Privacy-enhanced Microblogging

Vom Fachbereich Informatik der
Technischen Universität Darmstadt genehmigte

Dissertation

zur Erlangung des Grades
Doktor-Ingenieur (Dr.-Ing.)
von

Dipl.-Wirtsch.-Inf. Marius Senftleben

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Erstreferent:	Prof. Dr. Stefan Katzenbeisser Technische Universität Darmstadt
Korreferent	Prof. Dr. Frederik Armknecht Universität Mannheim
Tag der Einreichung:	17. April 2018
Tag der mdl. Prüfung:	5. Juni 2018
Hochschulkennziffer:	D17

Darmstadt 2018

Abstract

Microblogging is a popular form of Online Social Networking (OSN) activity. It allows users to send out short and succinct messages in a one-to-many publish-subscribe manner. Most current service providers are centralized and deploy a client-server model with unencrypted message content. As a consequence, all user behavior can, by default, be monitored, and censoring based on message content can easily be enforced on the server side. Also, the centralized services can be seen as a single point of failure and can become severely disrupted either by disconnecting the service from the Internet or by Internet outages. These privacy issues and the susceptibility of the services to censorship are not only theoretical, but a variety of incidents have shown that these issues are readily exploited. As a consequence, it would be desirable if alternative microblogging solutions were available that offer additional privacy features and which are more resilient to censorship. In this thesis, a distributed, peer-to-peer based microblogging system consisting of mobile smartphone-equipped users that exchange messages, encrypted under a group key, in an anonymous and censorship-resistant manner is proposed, which alleviates the privacy and censorship issues of most current centralized microblogging architectures. Users conduct peer synchronizations to exchange messages, i.e., the bidirectional exchange of a prepared message buffer whenever two users are in proximity of each other and able to establish a direct link using local communication means such as Bluetooth. The way in which the send buffer is filled with messages is determined by the use of a given synchronization strategy. To show the feasibility of the proposed microblogging system, we experimentally evaluate the message spread with simulations that run on a wide range of synthetic and real-world mobility inputs. We show that such systems are working for a range of mobility and network settings, and we evaluate the effects of using different synchronization strategies. In addition to a normal operation of the network, we also test it under adversarial conditions, e.g., under the presence of users which jam the network or send spam. Since the message propagation in the purely peer-to-peer based system is dependent on the mobility the users exhibit as well as the number of users encountered, which are needed to conduct peer synchro-

nizations, the message spread can be slow to non-existent, if parts of the network are segmented and users can not exchange messages via peer synchronizations. In such cases, the devised solution for a higher message spread is facilitated by employing a server that stores the messages of multiple groups in an Oblivious RAM (ORAM) data structure which can be accessed by users at their own discretion. On demand, users read or write their group-encrypted messages obliviously, without the server getting to know anything about the accesses that took place by the users including the message accessed or whether the access was a read or write access. In total, the microblogging solution's use is feasible, but the design decisions required for anonymity result in a delayed spread of messages on a best effort basis.

Zusammenfassung

Microblogging ist eine beliebte Form der Kommunikation im Bereich der Online Social Networks (OSN). Es erlaubt den Benutzern kurze, informative Nachrichten an eine Gruppe Interessierter zu versenden. Die meisten zur Zeit existierenden Diensteanbieter verwenden eine zentralisierte Client-Server-Architektur und versenden die Nachrichten unverschlüsselt. Eine sich daraus ergebende Problematik ist, dass das gesamte Nutzerverhalten überwacht werden kann und dass das serverseitige Zensieren von Nachrichten einfach durchgeführt werden kann. Zusätzlich ermöglicht die Zentralisierung des Dienstes Angriffsmöglichkeiten gegen die Verfügbarkeit, indem er vom Internet abgetrennt wird oder indem er unerreichbar durch Ausfälle des Internets wird. Diese angesprochenen Kritikpunkte hinsichtlich Privatheit sowie die Anfälligkeit für Zensur sind nicht nur eine theoretische Möglichkeit, sondern wurden durch eine Reihe von Vorfällen zur praktisch durchführbaren Realität. Aus diesen Gründen wäre es wünschenswert, alternative Microblogging-Dienste zur Verfügung zu haben, die ein Mehr an Privatheit bieten und zugleich resilienter gegenüber Zensurmaßnahmen sind. In dieser Arbeit wird ein verteiltes, peer-to-peer basiertes Microblogging-System vorgestellt, welches aus mobilen Smartphone-Nutzern besteht, die ihre Nachrichten unter einem Gruppenschlüssel verschlüsselt anonym und zensurgeschützt verbreiten können. Die Nutzer benutzen für das Verbreiten der Nachrichten Peer-Synchronisationen, d.h. es erfolgt – wann immer sich aufgrund der örtlichen Nähe zweier Nutzer die Möglichkeit ergibt einen direkten Kommunikationskanal zwischen den Smartphones aufzubauen – ein beidseitiger Austausch von vorbereiteten Nachrichtenpuffern. Die Art auf die die Nachrichtenpuffer gefüllt werden hängt dabei von einer gewählten Synchronisationsstrategie ab. Um die Durchführbarkeit des vorgeschlagenen Microblogging-Systems vorzuführen, wird die Nachrichtenverbreitung mittels Simulationen experimentell untersucht, die auf Grundlage eines breiten Spektrums von synthetischen und empirischen Bewegungsmustern der Nutzer erfolgen. Es wird gezeigt, unter welchen Bedingungen hinsichtlich Bewegungsmuster und Netzwerkeinstellungen die Nachrichtenverbreitung gelingt, inklusive der Auswirkungen von verschiedenen Synchronisationsstrategien. Neben dem ungehinderten Ablauf

wird dabei auch der Betrieb unter dem Vorhandensein von Gegenspielern getestet, die das Netzwerk durch Jamming oder das Versenden von Spam unbrauchbar machen wollen. Da die Nachrichtenverbreitung in einem reinen peer-to-peer basierten System von der Bewegung der Nutzer und den daraus resultierenden Begegnungen derselben abhängig ist, kann es vorkommen, dass die durch Peer-Synchronisationen erreichte Nachrichtenverbreitung langsam erfolgt oder sogar nicht stattfinden kann, falls Teile des Netzwerks segmentiert und somit voneinander abgeschottet sind. In diesen Fällen ist es vorgesehen einen höheren Verbreitungsgrad der Nachrichten durch die Benutzung einer Server-Komponente zu erreichen, die die Nachrichten der verschiedenen Gruppen in einer Oblivious RAM (ORAM) Datenstruktur speichert. Dieser Speicher erlaubt es den Nutzern lesend oder schreibend auf ihre Nachrichten zuzugreifen, ohne dass der Server aus diesen Zugriffen irgendwelche Informationen über die Art des Zugriffs oder auf die zugegriffenen Nachrichten gewinnen kann. Das vorgestellte Microblogging-System ist nutzbar, jedoch ist die Nachrichtenverbreitung aufgrund der für die Anonymität nötigen Designentscheidungen verzögert.

Acknowledgments

First of all I woud like to express my gratitude and thanks to my thesis supervisor Professor Stefan Katzenbeisser for accepting me as PhD student on his team and who thus gave me the opportunity to work on this thesis. He constantly guided and encouraged me throughout the work on this thesis and always managed to have time for discussions and questions. I would like to extend my thanks to Professor Frederik Armknecht for agreeing to be the second referee of this thesis.

I had a good time working on the SecEng team and would like to thank all the colleagues with whom I had the pleasure to meet over the years. During my time at the chair I experienced a friendly and productive atmosphere which was great for working on a project such as a dissertation.

Last but not least I wish to thank my family for their support and encouragement and for helping me not to lose faith in the completion of this thesis.

Contents

Chapter 1

Introduction

In this introduction at first a summary of the variety of Online Social Networking sites is given in Section 1.1. Then a short introduction of microblogging is presented and the need for privacy-enhanced microblogging is motivated. This is followed by an outlook of the remainder of the thesis in Section 1.2 and an overview of the publications that were made during the course of this dissertation in Section 1.3.

1.1 Online Social Networking

After the Internet's widespread growth in the 1990's, a new form of communication platforms emerged in the 2000's, the social networking sites (SNSs) or online social networks (OSNs). The sites that offer OSN services can range from weblogs, i.e., web sites that chronologically display postings of mostly persons but also of organizations, to social networks, i.e., web sites that allow registered members with a personal profile page to connect with other members of the social network and exchange messages. Geolocation services allow people with a mobile device, e.g., a smartphone, to broadcast their current location to their list of contacts, thus allowing people to meet based on the notices received from the geolocation service.

1.1.1 Microblogging as Online Social Networking Service

Microblogging is a popular form of Online Social Networking activity. It is part of an area of OSN known as micromedia, where small snippets of media, e.g., text, images, or video, are shared amongst users. Users usually can subscribe to the message channels of other users or create message channels themselves. *Twitter*[1] is the most prominent service used extensively for microblogging, with over 300 million monthly active users[2]. The name Twitter is derived from the verb *to tweet* and there

[1] `https://twitter.com`
[2] `https://about.twitter.com/company`

exists a terminology of its own for the user actions and users, with for example the following terms:

- Tweeter: the user who sends a short message of around 140 characters.
- To tweet: the user action of sending a message.
- Tweet: a microblogging message.
- To re-tweet: the resending of a tweet of a user under its own name.
- Follower: a user that subscribes to all tweets sent by a tweeter.
- Hashtag: the sign # is used to indicate search terms and categorizes tweets.

1.1.2 Privacy and Censorship Issues in Microblogging Services

Microblogging services allow users to send out clear, succinct and informative messages of around 140 characters to their subscribers.

The communication in the offered services is typically in plaintext and public; furthermore, all widely adopted services (such as Twitter) follow the client-server model. The interaction with Twitter is usually done via a web interface or an app for smartphones, so that the sender (tweeter) logs in either using the Internet directly or by indirectly connecting to it via the mobile telephony network. The sender can then post (tweet) messages on his named user channel as he seems fit. In order to tag a message with a specific topic the hashtag-character can be placed directly before a term. Subscribers (followers) can receive all the messages (tweets) from the sender, if they have subscribed to a certain user channel. The sum of all channels of wich a user is a follower are the user's subscriptions. Also, users can forward messages under their own name (re-tweet). Finally, users can search for messages using the hashtags assigned to the messages as query terms. Even if transport encryption is used, all user behavior is transparent on the server-side.

Unfortunately, these design decisions create numerous privacy and security problems, particularly in more oppressive political surroundings.[3] For example, Privacy International, a privacy rights group, has obtained evidence that intelligence agencies of the United Kingdom are collecting social media information on millions of individuals with mass surveillance programs. Large searchable databases contain bulk personal datasets (BPDs) per individual which can include "details such as an individual's religion, racial or ethnic origin, political views, medical condition, sexual orientation, and legally privileged, journalistic or 'otherwise confidential' information". These BPDs "vary in size from hundreds to millions of records", and

[3] Cf. reports of reporters without borders, `https://rsf.org`.

are acquired by "overt and covert channels".[4] Another example is a program named *PRISM* which allows United States intelligence agencies to directly tap into central servers of leading Internet companies, with access to text, audio and video chats, e-mails, documents and connection logs.[5]

Since most of the existing microblogging services do not hide the message content from the service operator, confidentiality is not addressed.[6] Thus, all content generated by the users can be analyzed by the service providers. In addition, the complete user activity is transparent to the providers, such as messages sent and received by users or each user's group subscriptions. Moreover, all user interactions are known to the central provider, among them all messages sent, all existing subscriptions, the entire query-patterns of users, etc. Complete data retention facilitates traffic analysis as well as data mining on the unencrypted messages. Incidents indicate that such data retention is routinely done in practice. For example, a program named *Tempora* allows British intelligence agencies to set up a large-scale database that buffers and stores internet communication for 3 days and metadata that describes basic information on who has been contacting whom for 30 days. The key source for obtaining the data are intercept probes attached to transatlantic fibre-optic cables, essentially capturing the traffic flowing between the Internet exchange points of Western Europe and North America.[7]

Current microblogging services are also prone to censorship. All messages are relayed over central servers and messages can be censored based on their content and sender or receiver, thus prohibiting the spread of content according to censorship policies. Due to the centralized nature of the services and the messages in plain text, acts of censorship can easily be performed either by the service providers themselves or by external parties. Furthermore, centralized systems are vulnerable to local Internet disruptions and outages, either unintentionally, e.g., power grid failures, or intentionally, e.g., blocking specific services or Internet shut-downs via 'kill-switches'. The intentional censorship that takes place can on the one hand only target a limited number of OSN services or specific Internet sites and studies show that a multitude of services and websites such as Twitter, YouTube, Wikipedia or Google have been blocked in parts of the world. [119] On the other hand, complete Internet shut-downs have also been reported on numerous occasions.

[4] Cf. https://techcrunch.com/2017/10/17/uk-spies-using-social-media-data-for-mass-surveillance/.

[5] Cf. https://www.washingtonpost.com/investigations/us-intelligence-mining-data-from-nine-us-internet-companies-in-broad-secret-program/2013/06/06/3a0c0da8-cebf-11e2-8845-d970ccb04497_story.html.

[6] Of course it can be conceded that fast message broadcasting and not confidentiality was the design goal and therefore the comparison might not be adequate.

[7] Cf. https://www.theguardian.com/uk/2013/jun/21/gchq-cables-secret-world-communications-nsa.

An earlier example of such an Internet shut-down took place in Burma in 2007, when a sharp increase in fuel prices sparked protests in the population. Using cell phones and the Internet a small group of Burmese protesters circulated information amongst themselves. Also, they created awareness outside of Burma by photographs and videos sent over the Internet. In response, the Burmese military made use of its full control over the country's Internet gateways and shut down the Internet access completely in order to stop these information flows. Additionally, the cellular telephony infrastructure was reportedly disrupted.[8] Another example can be found in the wake of protests and riots in numerous North African and Middle Eastern countries that started in late 2010. Denial of Internet access was a strategy used by authorities to prohibit demonstrations and to prevent information to be disseminated via the Internet. Egypt and Libya experienced full Internet shut-downs in attempts of the governement to quell protests [33]. The most aggressive form that was also used in Egypt consisted of the removal of the countries Border Gateway Protocol (BGP) entries. BGP exchanges information on active IP address ranges between the Internet service providers all over the world, thus facilitating connections between the multiple autonomous networks worldwide. However, the deletion of the 3,500 Egyptian prefixes meant that this routing information was missing from the routing tables of BGP routers around the world. Consequently, routers could no longer send packets to an IP address that fell within these Egyptian prefixes. The authorities had issued the order to disconnect from the BGP to the countries four Internet service providers, and they complied with the order, thus effectively removing the country from the Internet.[9]

In both cases – unintentional Internet outages due to power grid failures and intentional Internet shut-downs – most current microblogging systems cease to function.

1.1.3 Need for Privacy-enhanced Microblogging

These aforementioned problems call for alternative microblogging architectures that are privacy-friendly. In this thesis, a private mobile microblogging architecture is proposed that respects the users' privacy and is resilient to censorship. In this thesis the focus is laid on microblogging of text messages. In particular, we rely on the fact that smartphones are becoming ubiquitous communication devices, which are equipped with local communication links (such as NFC links and ad-hoc Wifi networks), while having Internet connectivity. The smartphones can thus be a part of a mobile system such as a Vehicular Ad-hoc Network (VANET). Instead of relying on a centralized infrastructure to exchange messages, the proposed solution is based on a Peer-to-Peer (P2P) architecture, involving mobile peers that exchange messages

[8] Cf. `https://opennet.net/research/bulletins/013`.
[9] Cf. `http://techland.time.com/2011/01/28/how-egypt-cut-off-the-internet/`.

with each other using local radio links and a message synchronization strategy, an action named peer-sync which does not rely on any form of Internet or wide area network access. This peer synchronization essentially allows peers to microblog short messages privately using their smartphones while on-the-go. Optionally, users can also upload messages to a server or download messages from a server by an action called server-sync. These server synchronizations are optional and can be enabled or disabled by the users. Messages are transmitted in encrypted form so that only a group of authorized peers can access it; messages that cannot be read by one peer will nevertheless be forwarded in a re-randomized fashion in order to guarantee appropriate message distribution, while at the same time providing unlinkability, i.e., an adversary cannot identifiy if messages are related or not, against passive adversaries that do not actively participate in the network and only observe network traffic.

Moreover, the proposed architecture offers censorship-resistance: The distributed data storage at all peers contains multiple replications of messages and is thus difficult to censor. The decentralized message dissemination over point-to-point links conducted by dynamically moving peers is hard to shut down in its totality, thereby increasing the availability. The use of randomizable encryption allows to gain unlinkability of messages as well as sender anonymity, i.e., an adversary cannot sufficiently identify the sender within the set of the users. Finally, the use of (group) encryption limits the number of parties who can access messages.

1.2 Outlook

The rest of this thesis is structured as follows. Chapter 2 surveys related work, ranging from early anonymity systems to solutions focusing on privacy-enhanced microblogging. At first the chapter looks at the main anonymity systems, namely mixes, mix networks, onion routing and The Onion Router (TOR) as well as crowds. An overview of anonymity in mobile networks is then given, followed by a more detailed look on solutions providing anonymity for microblogging in different network settings, including a look on some privacy-friendly instant messaging apps.

Chapter 3 details the envisioned privacy-enhanced microblogging solution. The chapter firstly introduces the required anonymity terminology and states different adversary capabilities, before the desired privacy and security goals of the privacy-enhanced microblogging solution and the assumed adversary model are described. Then, the architecture of the privacy-enhanced microblogging solution with its core components is introduced, including the infrastructure, the message format and message storage, the used encryption scheme, group and key management, and the synchronization of messages using the two mechanisms of peer-syncs and server-syncs. An analysis of the extent to which the stated goals of privacy, anonymity

and censorship-resistance can be met is then made, followed by some remarks on the limitations of building an anonymity system in general. Finally, the chapter includes a short outlook on a built proof-of-concept demonstrator which shows how an actual implementation of the microblogging solution can look like.

Chapter 4 shows the feasibility of the proposed microblogging system using a simulation running with peer-syncs only, i.e., no Internet or wide area network access. It explores and tests the boundaries in which the microblogging system works. To that end, different empirical mobility datasets and a wide range of synthetically generated mobility datasets are created. The setup of the simulations to which the datasets serve as input is then described and explained, before the simulation results for the runs without adversary interference are presented and analysed. Subsequently, adversary-tampered simulation runs and their effects with respect to anonymity and censorship-resistance are examined, before a short discussion on the limitations of the proposed microblogging system is given.

Chapter 5 looks at an extension of the peer-sync process using private set intersection protocols. The chapter first reviews existing classes of private set intersection protocols and then details one promising protocol that is well suited for the given application scenario. The chapter then describes simulations of the peer-syncs using private set intersection and analyses the obtained results.

In Chapter 6 an oblivious random access memory (ORAM) extension for the server-syncs is introduced. We require this extension, because in cases of network segmentation in the peer-to-peer network a server enables to distribute messages accross segments. The server is for privacy reasons required to not be able to deduce any group memberships of the users. For this purpose, an ORAM that does not demand expensive computations from the client is chosen and adapted to the given application scenario. With the found solution the clients do only need to be online when they want to access the server, and the privacy of the clients access patterns against the server is guaranteed at all times.

Chapter 7 concludes this thesis.

1.3 Publications

This dissertation is based on the following publications:

- *Microblogging in a Privacy-Preserving way* with Nikolaos P. Karvelas and Stefan Katzenbeisser, in 12th International Workshop on Frontiers in Availability, Reliability and Security (FARES 2017), August 29–September 1, 2017, Reggio Calabria, Italy. [68]

- *On the Privacy and Performance of Mobile Anonymous Microblogging* with Ana Barroso, Mihai Bucicoiu, Matthias Hollick, Stefan Katzenbeisser and

Erik Tews. In: IEEE Transactions on Information Forensics and Security, 11(7):1578–1591, 2016. [112]

- *MoP-2-MoP – Mobile private microblogging* with Mihai Bucicoiu, Erik Tews, Frederik Armknecht, Stefan Katzenbeisser and Ahmad-Reza Sadeghi. In: 18th International Conference on Financial Cryptography and Data Security (FC 2014), March 3-7, 2014, Christ Church, Barbados. [113]

Chapter 2

Related Work

Related work is first outlined in Section 2.1 on anonymity systems in general, then a review on anonymity in mobile networks is given in Section 2.2, before related work on privacy and anonymity in microblogging systems is investigated in Section 2.3. Finally, a conclusion on the studied related work is given in Section 2.4.

2.1 Anonymity Systems

For web traffic some common anonymity systems include mixes, onion routing with Tor, and crowds, concepts which we subsequently describe.

2.1.1 Mixes and Mix Networks

One of the first ideas for the construction of an anonymity system is that of a mix proposed by Chaum [22]. A mix has an asymmetric key pair of which the public key is known to senders. Senders encrypt their message alongside with some randomness and the identity of the recipient of their message under the public key of the mix. The mix stripes off the encryption using its corresponding private key and waits until a threshold amount of messages has arrived. The mix then sends all the accumulated messages out to their recipients in a different order than they arrived at the mix. This kind of mix is called a threshold mix and the flushing, i.e., the re-sending of messages, is done once a fixed amount of messages has arrived at the mix. Other flushing strategies of a mix can be the re-sending of all messages that arrived in a specified time span, i.e., a timed mix as proposed in [114], or a stop-and-go-mix [70], i.e., mixes which delay messages by a user-specified time that is appended to the packet before encrypting it with the mixes public key. The different flushing strategies have all been devised with the aim to achieve unlinkability in the face of a global passive adversary, who should not be able to link the incoming and outcoming messages of the mix.

Mix networks extend the idea of one mix to a chain of mixes. Messages are relayed over multiple mixes that pool the messages and send them over to the next mix until they reach their receivers. Each mix has an asymmetric cipher key pair of which the public key is known to the senders. A user can either configure a route of mixes over which to relay his messages himself, or he can make use of pre-configured routes to generate a mix cascade. The message must be encrypted in multiple layers, so that the innermost encryption is done using the public key of the last mix of the route and the outermost encryption is done using the public key of the first mix of the route. The encryption of the message of the intermediate mixes are done accordingly. Each mix removes the outermost encryption layer, then the messages are sorted into a new permutation and send along to the next mix in the route according to a given flushing strategy as described above.

2.1.2 Onion Routing and Tor

Onion routing [55] is a circuit-oriented adaption of the layered encryption of mix networks whose goal it is to avoid high latencies caused by the pooling of messages in mixes before they are flushed. The aim of getting closer to real-time traffic is obtained by omitting the mixing, i.e., the pooling of messages and flushing according to a given strategy. Users select a circuit of relay nodes, i.e., entry node, intermediary nodes and exit node, over which the layer-wise encrypted packages are routed, and each node only knows its predecessor and successor. All traffic is directly forwarded over these circuits, thus minimizing delays. Onion routing is the basis of Tor [39], the reference implementation of the onion routing concept. The Tor project[1] is tailored for web traffic and allows to route TCP streams over a circuit of relay nodes. Tor's basic approach does not guarantee anonymity even against a global passive adversary, and a number of attacks on the Tor network have been investigated [7, 51, 88]. Yet, Tor with its easy to use client and interface is a popular and widespread used anonymity tool [27], which is actively maintained and improved [24, 61, 93].

2.1.3 Crowds

An early peer-to-peer approach for anonymity in web transactions is Crowds [104, 105]. Peers probabilistically create a static path over which web transactions are relayed.

The crowd is a dynamically formed user base. To connect to other users and become part of the crowd, a user has to connect to a contact server which has to admit the user's request to join the crowd. Once a user joined the crowd, the contact server sends the needed key material to the newly added user, since transport encryption

[1] `https://www.torproject.org`

established between each pair of users is required by the crowd mechanism. The server reports to the joining user also the current user base of the crowd so that he can connect to other users. Each user of the crowd thus has to be a member of the crowd. When the user wants to start a web session, a route involving multiple other users is probabilistically created as follows: Randomly, the user picks another user of the crowd and forwards his encrypted request to it. The receiving user forwards the request newly encrypted to another randomly chosen user with probability $1 > p_{forward} > 0.5$, otherwise with probability $1-p_{forward}$ the request is forwarded to the destination server. Once a path is established further messages from the initiating user are relayed over the same route to the server, and the server's responses are relayed over the same route to the initiating user. With this construction a sender anonymity of "beyond suspicion" is achieved, i.e., even though the adversary is aware that a message has been sent, the sender is from the perspective of the adversary no more likely to be the originator of that sent message than any other user in the system, since every user relays requests to a randomly chosen other user of the crowd; thus for each user the probability of being the original sender is the same.

2.2 Anonymity and Routing in Mobile Networks

Solutions that address anonymity in mobile ad hoc networks (MANETs), where routing between mobile devices is done in a self-configured manner, include ALARM [37] and MASK [137]. ALARM is a secure link-state based routing protocol which achieves anonymity and untraceability of the participating nodes, as well as other security properties such as data integrity by leveraging group signatures. In order to operate, the nodes' current locations are securely disseminated to construct topology snapshots and to forward data. However, each node needs to periodically broadcast its location to all other nodes by flooding, making scalability an issue for larger MANETs containing thousands of nodes. MASK is an anonymous on-demand routing protocol, which achieves anonymous MAC-layer and network-layer communications. It is based on the use of dynamic pseudonyms instead of static MAC and network addresses. The protocol achieves sender and receiver anonymity. Node unlocatability is also an included goal, meaning that although adversaries might know some real network addresses, they are unable to decide where the corresponding nodes are in the network. In the context of vehicular ad hoc networks (VANETs), where revocable anonymity is needed for liability issues, pseudonymity schemes are typically used to provide anonymity during normal operation [44].

In delay-tolerant networks (DTNs), which maintain no explicit routing information, human mobility and its characteristics (cf. [106]) are leveraged for message propagation. Su et al. [125] argue, based on collected human mobility data, that effective routing decisions can be made by only knowing the pair-wise contacts that

took place between nodes, irrespective of mobility models or location information. Specifically, they claim that human mobility traces allow devices to beneficently route packets leveraging short-range radio to extend the range of existing infrastructure or to provide a means to communicate in places where there is no existing infrastructure. Chaintreau et al. [20] empirically demonstrate, that the distribution of the intercontact time[2] between wireless devices carried by humans can be approximated by a power law distribution for durations ranging from 10 minutes up to a day. One of their findings was the discovery of a new class of message forwarding algorithms named "oblivious forwarding algorithms", which are in contrast to other findings well approximated by a power-law distribution. In Humanets [4], smartphone-to-smartphone communication is used to more efficiently propagate messages, while at the same time avoiding the use of mobile telephony networks. In the approach used in this thesis, these observations are leveraged both in the microblogging system and in its simulation. A number of solutions for content distribution in DTNs have been proposed (cf. [84], [87], [86]). However, propositions focusing on anonymity in such scenarios are scarce: Rogers et al. [108] focus on secure communication over diverse networks achieving confidentiality, authenticity, integrity and forward security, but do not specifically address anonymity. The property of forward secrecy is gained by using secrets initially shared between all endpoint pairs of devices, from which a one-way key derivation function creates short-lived temporary secrets. In contrast to these approaches, the microblogging system used in this thesis targets strong privacy and anonymity properties.

2.3 Anonymity for Microblogging

In this section related work with respect to microblogging is discussed. Firstly, this is done in Section 2.3.1 for client-server solutions that by design are prone to system outages and censorship. Then, in Section 2.3.2 solutions that deploy techniques from peer-to-peer technologies in a wide are networking setting similar to the Internet are discussed, followed by solutions tailored towards a delay-tolerant networking setting in Section 2.3.3. To complete the related work section a brief look on instant messaging applications for smartphones is done in Section 2.3.4.

2.3.1 Client-Server Solutions

Privacy-enhanced client-server microblogging similar to Twitter is proposed in [30], describing a microblogging server that matches encrypted messages to subscribers using oblivious matching, resulting in a Twitter-like service with two particular privacy

[2] Chaintreau et al. define this term as the time gap separating two contacts between the same pair of devices.

goals for tweeters and followers: first, a tweeter encrypts all his tweets and is able to choose who is allowed to access them via defining an access-control list based on the tweets' content, i.e., their hashtags. This enables low-level content based authorization of followers. Second, followers can subscribe to arbitrary hashtags and by doing so do not leak any information thereof to any other entitiy, resulting in privacy for followers. In order to guarantee these privacy goals, the system relies on a trusted third party called Hummingbird server, to which all users register in order to get their personal RSA key pair. By the use of an Oblivious Pseudo Random Function-protocol [46], the privacy of followers is ensured when issuing a follow request for another tweeter, i.e., the content of the following-by-topic hashtags are hidden from the Hummingbird server. The matching of tweets to followers is effectuated by the Hummingbird server using oblivious matching. Thus tweets are matched to followers while learning nothing about the cleartext tweets and hashtags. The adversary is assumed to be honest-but-curious, i.e., it does not deviate from the specified protocols. Yet, anonymity and censorship-resistance are no design goals.

The Twitsper server [118] is, similar to the Hummingbird server, added to the existing Twitter architecture. Twitsper extends Twitter's API with three functions: privately send a tweet to all users in a specified group; determining whether a tweet is private; and the option to privately reply to all the tweeter's followers who received an original tweet. The Twitsper server integrates into Twitter and acts as the facilitator of communication amongst private groups, thus enabling the private sending of tweets only by handling the associated metadata (the tweets are still handled by Twitter's servers). The privacy gains are limited compared to those of the Hummingbird Server: users who are not member of a private communication should not be able to infer which users are involved in the private communication, and the Twitsper server itself should also not be able to infer any member of private communications. In addition, the server should also not know the size of a private group. Anonymity and censorship-resistance are again no design goals.

To achieve anonymity in these scenarios, the use of Tor or a subscription service with unlinkability of users across logins such as [75, 76] is needed.

A different approach is followed in h00t [5]. Their idea is to take the existing Twitter infrastructure with its numerous Tweets as cover traffic and to overload Twitter's hashtag mechanism to create censorship-resistant microblogging. h00t provides an interface that is much like Twitter, except that the original plaintext hashtags are replaced with short hashes of them, followed by the encrypted message. Simplified, every Tweet consists of the AES-encrypted message text prepended by the only 24 bit long hash of the original plaintext hashtag. In this way, the short hashtags from different original plaintext hashtags are meant to collide. Each user interested in a specific hashtag has to download all the messages that have the same short hashtag

as the actual original plaintext hashtag he is interested in. The authors claim that this achieves a form of censorship-resistance, since only a so-called heavy-handed censorship can be applied by the adversary: a specific original plaintext hashtag can only be censored through its short hash, creating the necessity to censor all the other original plaintext hashtags that have the same colliding short hashtag, i.e., no fine-grained censorship is possible. The recipients in this way also obtain a form of receiver anonymity, since they have to download a larger amount of messages than the ones they are actually interested in. These additional messages serve as cover traffic and the h00t client software has to decrypt and filter out all the superfluous messages downloaded. However, the possibility of censorship, in particular heavy-handed censorship, based on the deployed client-server infrastructure used by Twitter still persists.

Twitterize [36] also relies on the pre-existing client-server architecture of Twitter. Twitterize's goal is to provide confidentiality and anonymity to the users. To achieve confidentiality all tweets are encrypted. For anonymity overlay networks are created for each hashtag, which consist of subsets of all the Twitter users that make up the network. All overlays contain additional users referred to as forwarders which are not interested in tweets of the given hashtag. Their function is not to send or receive tweets for this given hashtag, but to increase the size of the anonymity set. The overlay networks connect senders and receivers without the requirement to directly communicate with each other. The system is built in such a way that senders and receivers are decoupled and that a singular forwarder is unable to link senders and receivers. As a kind of mix network this is a privacy-preserving peer-to-peer overlay constructed on top of the existing infrastructure of Twitter.

2.3.2 Peer-to-Peer Solutions in a Wide Area Network Setting

Some wide area network dependent peer-to-peer microblogging systems exist, but they rarely focus on anonymity and censorship-resistance.

One first decentralized microblogging solutions based on the peer-to-peer paradigm is Litter [65, 67]. Litter assumes the presence of a peer-to-peer middleware that enables direct communication amongst peers. In particular, they use SocialVPN [66], a peer-to-peer virtual private network which uses social networks in order to establish encrypted IP tunnels between befriended peers. On top of these encrypted links they build a decentralized microblogging approach with best-effort UDP multicast packet delivery, i.e., one that does not use acknowledgments or timeouts for message retransmissions. They propose a push/pull mechanism for propagating messages, a message format, and a message revocation mechanism. The solution thus offers the benefits of decentralization in a peer-to-peer solution with best-effort propagation. The security guarantees are however limited to link encryption between peers

based on public-key cryptography in conjunction with a public key infrastructure. Anonymity and censorship-resistance were no design goals.

Megaphone [94] is another solution that targets fault-tolerance and scalability through decentralization. They also argue that the current web-based client-server services for microblogging introduce a single point of failure. The peer-to-peer technology used by Megaphone is Pastry [109], a peer-to-peer middleware that organizes a network through nodeIds which are based on a hash of unique characteristics of a nodes device, such as MAC address or IP address, on top of which Scribe [19] is used, a scalable application-level multicast infrastructure that supports large numbers of groups, also with large number of members per group. Megaphone itself makes use of a public-key infrastructure so that messages can be signed and encrypted, and the Pastry's nodeIds are used together with Scribe to distribute the messages per group in the peer-to-peer network.

In [43] a cryptographic mechanism is presented, which allows a peer to anonymously request messages from other peers in a peer-to-peer network. In this network setup peers are only connected to personally trusted peers, and a peer forwards his message directly to its trusted peers. If a peer missed a message, a use of hierarchical identity based encryption takes place, so that this peer can pull the missed messages from the other trusted peers while maintaining anonymity of the peers involved.

Yet, all approaches need Internet access, in contrast to the scenario pursued in this thesis, and the security guarantees exclusively either focus on anonymity or censorship-resistance, but not on both.

2.3.3 Delay-Tolerant Networking Solutions

Few microblogging systems in delay-tolerant networks, as used in the proposed system, exist. In the area of ubiquitous computing a system named floating content [91] foresees an ephemeral content sharing service, built and wholly dependent on the mobility of users with their mobile devices. Any given user is able to create content, which is only created locally, tagging it with its geographic origin alongside with a validity radius and an expiration time. These three items define the anchor zone in which the content is disseminated and for how long it should be disseminated. Dissemination of the content is first achieved by the creator's device to its neighbors within the validity radius, and then the dissemination is organised by the other nodes within the anchor zone, given that there are enough participating mobile devices in the anchor zone. If this is the case, the content floats in that fixed zone, and if too few nodes are in the zone, the content vanishes. In total, this geographically limited content dissemination is simulated so as to see the possible lifetimes of the content, and criticality conditions have been derived [126] for this ubiquitous computing use case.

In a network setting similar to ours, Robin [8] proposes a microblogging system that uses community reputation to prevent flooding of the network with junk messages. A flooding attack is done by malicious users that create junk messages on behalf of other users and of Sybil users with fake identities. Their goal is a Denial-of-Service attack by propagating their junk messages to as many other nodes as possible. A technique called diversity selection treats the problem of selecting the reliable messages out of a pool of messages which contains far fewer reliable messages from real users than junk messages created by malicious users. However, sender anonymity is not a goal since each message is associated with the public key of the creator. Also, nodes need to choose a number of trusted nodes from the social graph called entry points. The possible flows of messages in the social graph from the creator to the entry points are then investigated, and it is claimed that a high number of junk messages is routed along the attack edges from malicious users. Flow control is then enforced to limit that number of junk messages. The idea of Robin is continued in the work of Rangzen [77], which proposes a mobile microblogging solution with trusted message propagation by the use of social graphs and additionally private-set intersection protocols to prioritize messages.

Yet, the foci of these works are not on anonymity and censorship-resistance, and no other solutions in this niche are known to the author.

2.3.4 Private Instant Messaging Apps

This section briefly deals with instant messaging apps for smartphones, since they are to some extent related to microblogging. Their main difference is, that they mainly focus on one-to-one communication, but they do also offer one-to-many communication. Messages get delivered instantly, just as in texting, which has since the advent of widespread mobile telephony in the mid to late nineties been a popular form of communication. To achieve real-time communication, the messaging app clients are constantly in contact with a server. There are only instant messaging apps that work with the client-server paradigm and no peer-to-peer solutions. One of the most popular apps is called *WhatsApp*[3], and even though after initial criticism end-to-end encryption has been introduced, the app does not have all the features needed to be labeled privacy-friendly. This is because the provider still has access to the encryption keys used and can thus break the confidentiality of the messages exchanged between users. Another issue is that there is no mechanism that allows users to verify their correspondents identity[4].

But there are other instant messaging apps for smartphones that are more focused on security and privacy aspects.

[3] `https://www.whatsapp.com/`

[4] Cf. `https://www.eff.org/secure-messaging-scorecard`

TextSecure[5] is one app that uses AES encryption and the keys used to encrypt the messages are stored exclusively on the users device. Also a passphrase can be used to encrypt the entire local message database and the secret keys used to communicate privately with the other users. Each user gets assigned a unique fingerprint which can be communicated over the phone by voice, but a verification is also possible upon personal encounter of two users by generating a barcode of the user fingergrint that is displayed on the device and scanned with the other users smartphone. The apps source code is freely available.

Threema[6] is another app with end-to-end encryption. Users receive a random user id which is not linked to their email address or phone number, thus allowing to use the app with a certain degree of anonymity. Asymmetric ECC key pairs are generated by and used with the open source cryptography library NaCl[7] to encrypt and decrypt the messages. To verifiably obtain another user's public key a user can scan its QR code when they personally encounter each other. The contact lists are managed on-device only and the messages are deleted from the servers immediately once they have been delivered.

2.4 Conclusion

This chapter outlined the main anonymity systems designed for the Internet, e.g., mixes, mix networks, and crowds. The two approaches of mix networks and crowds (peer-to-peer) have been discussed extensively [14], and ultimately the layered encryption of mix networks has been further developed to onion routing. The Tor anonymity network runs based on onion routing and is currently the most widespread system used for anonymity on the Internet.

The focus was then shifted to microblogging solutions with additional security and privacy features in different network settings: client-server, peer-to-peer with wide area networking, and delay tolerant networking. The review of this related work showed that the security and privacy guarantees of previous constructions solely focused either on anonymity or on censorship-resistance, but not on both at the same time.

[5] `http://whispersystems.org/`
[6] `https://threema.ch/en`
[7] `http://nacl.cr.yp.to/`

Chapter 3

Microblogging Solution

In this chapter needed anonymity terminology and adversary capabilities are introduced in Section 3.1, alongside with a definition of the solution's privacy and security goals – namely (sender) anonymity, privacy, and censorship-resistance – and an outline of the assumed adversarial capabilities. A description of the new microblogging solution is given in Section 3.2. This is followed in Section 3.3 by a discussion on how the desired privacy and security goals can be met by the proposed solution. Section 3.4 contains remarks on the difficulties of building an anonymity system in general. Finally, in Section 3.5 a built demonstrator of the microblogging solution is described.

3.1 Prerequisites

This section will first introduce some needed terminology surrounding anonymity, and then elaborate on adversary capabilities. The related notion of privacy deals with making a boundary between what information belonging to an individual belongs to the public domain and what information belongs to the private domain. If privacy is seen as the right to be left alone [129], then anonymity deals with the right to leave one's identity undisclosed.

This section first introduces the needed nomenclature for privacy and anonymity in Section 3.1.1, illustrates different capabilities of adversaries in general in Section 3.1.2, states this solution's design goals of anonymity, privacy and censorship-resistance in Section 3.1.3, and lastly outlines the adversary model assumed in this thesis in Section 3.1.4.

3.1.1 Anonymity Terminology

A step towards a standardization of the vocabulary in the field of anonymity research is the anonymity terminology initially proposed by Pfitzmann and Köhntopp [95, 96], from which some of our terminology is taken:

- *Anonymity*: anonymity of a subject from an adversary's perspective means that the adversary cannot sufficiently identify the subject within a set of subjects, the anonymity set.

- *Sender anonymity*: a sender is from an adversary's perspective anonymous within a set of potential sender, his/her sender anonymity set, which itself may be a subset of all subjects in the anonymity set who may send a message from time to time.

- *Unlinkability*: an adversary cannot sufficiently identify if items of interest, i.e., subjects, messages, or actions, are related or not. If linkability, the opposite of unlinkability, is given, the adversary can get insights on the connection between certain messages, their respective senders and receivers. Unlinkability is thus stronger than anonymity, since it comprises anonymity.

- *Undetectability*: an adversary cannot sufficiently distinguish whether an item of interest, i.e., subjects, messages, or actions, exists or not.

- *Unobservability*: undetectability of the item of interest, i.e., subjects, messages, or actions, against all subjects uninvolved in it and anonymity of the subject(s) involved in the item of interest, even against the other subject(s) involved in that item of interest.

3.1.2 Adversary Capabilities

The above-mentioned anonymity terminology shows that anonymity is achieved if an adversary can neither identify the sender nor the recipient of a message, i.e., the adversary can not determine who is sending to whom what specific message. These properties are named sender anonymity and receiver anonymity.

The capabilities of adversaries are, generally speaking, limited by the extent to which they can use the three resources people, time and both hardware and software equipment. However, a characterization of adversary capabilities with respect to the following distinctions is usually made [103, 133]:

- *Participation*: the adversary can be *internal* and be a part of the network, or not be a part of the network and be *external*.

- *Capability*: the adversary may either be *passive* and thus able to monitor and record traffic or *active*, thus being able to inject, drop and modify traffic, but unable to break cryptographic primitives such as encryption schemes (a so called Dolev-Yao-Attacker).

- *Mobility*: a *static* adversary can only monitor the same subset of the network, whereas a *dynamic* adversary chooses a subset of the network based on the information obtained from it.
- *Visibility*: a *global* adversary is able to access all network traffic, in contrast to a *local* adversary, who has only access to a subset of the network.

3.1.3 Privacy and Security Goals

Anonymity, privacy, censorship-resistance are the desired goals of the mobile private microblogging solution presented in this thesis:

Anonymity Sender anonymity is pursued; that is, an adversary should have no information on the originator of a message. This can also be achieved if message unlinkability is given, since it comprises sender and receiver anonymity. In our solution the desired property of anonymity is achieved by fulfilling the stronger property of unlinkability.

Privacy Group memberships of a peer and all messages should be kept confidential. This amounts to some form of receiver anonymity, since the adversary is unable to deduce which messages of all the messages received by a user are actually of interest to him, i.e., there is cover traffic.

Censorship-resistance No central entity should have the power to censor messages based on their content, and the message propagation should not be fully stopped by technical means. Simply put, this means that a message – once it has been created and relayed to other mobile devices participating in the network – can not be stopped from further propagation.

3.1.4 Adversary Model

An adversary model is considered in which there is a distinction between wide-area network (WAN) communications, e.g., for any communication conducted over the Internet, and local point-to-point communications deployed in the peer-to-peer (P2P) network. We assume that the adversary is *not* able to break cryptographic primitives.

The capabilities of the WAN adversary are:

- All WANs are under full passive control of the respective operators. The adversary can monitor and log all such communication channels used.
- Shut-downs of central communication infrastructures ("kill-switches") occur.

The capabilities of the P2P adversary are:

- Limited local jamming, monitoring or logging of radio links is possible.

- Malicious peers with full control of the communication channels used exist.
- Participating peers are not compromised by malicious ones.

With respect to the adversary properties from Section 3.1.2 the considered adversary in this thesis has the following capabilities: for WAN communications the adversary is an active one with global visibility and external participation. For the P2P communications the adversary is an active one with local visibility, dynamic mobility, and internal participation.

3.2 Architecture

The proposed microblogging solution *Mobile Peer to Mobile Peer (MoP-2-MoP)* builds upon mobile peers that interact with their smartphones using point-to-point communication links once they are physically close (technically they form an unstructured peer-to-peer overlay network). The requirements of this microblogging solution are the following:

- It should probabilistically propagate the messages issued by any user at some point in time in the absence of an attacker.
- It should provide the stipulated privacy and security goals of anonymity, privacy and censorship-resistance as defined in Section 3.1.3.
- It should be secure against the adversary modeled in Section 3.1.4.
- The microblogging solution should work with commodity-of-the-shelf hardware, i.e., smartphones, and should be as intuitively to use as possible.

The rest of this section yields the overview of the microblogging infrastructure in Section 3.2.1, followed by a description of its core functional components in Sections 3.2.2–3.2.5.

3.2.1 Infrastructure and System Overview

First a broader infrastructure overview is presented and then a more specific system overview is given in this section.

Infrastructure Overview

The considered microblogging system is a decentralized, distributed peer-to-peer network made up of mobile peers. The peers are humans carrying a mobile device (such as a smartphone). Specifically, the mobile peers are considered to form a dynamically changing peer-to-peer network, where the movement patterns correspond to

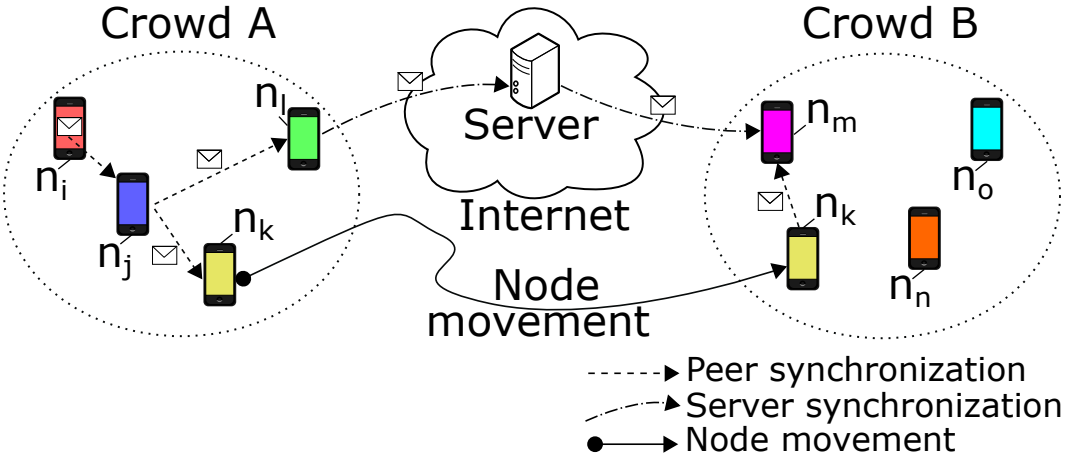


Figure 3.1: *Schematic overview of the microblogging infrastructure.*

the natural movements of the smartphone owners. All peers maintain a local buffer of encrypted messages. Whenever two peers are physically close to each other they exchange messages based on a fixed strategy (described in Section 3.2.5). By this local message exchange, the system aims at propagating new messages through the entire network. This process is referred to as peer synchronization. In case network segmentation occurs, peers can – as a backup solution – also download messages from servers using a wide-area communication network by an action called server synchronization; note that there can be multiple such servers, as they only serve as additional channel to transfer messages.

Figure 3.1 shows a schematic overview of the infrastructure. Peers are depicted by a smartphone-like shape, where n_{index} represent different peers. The dotted ellipses named Crowd A and Crowd B represent clusters of peers which are physically close to each other so that peer message synchronization is possible. Consider the following example: suppose that user n_i acts as originator of a message; as soon as n_i and n_j are close, they initiate a peer synchronization of their message buffers; this results in the new messages being spread. This way, the message sent by n_i finally reaches all other physically close peers in Crowd A (n_l and n_k in the figure) through peer synchronizations, depicted as dashed lines. The original encrypted message gets re-randomized at each hop so that a global adversary cannot link exchanged messages. A physically distant Crowd B can get the messages via two different mechanisms. First, a peer n_k can physically move close to a peer in Crowd B and initiate a peer synchronization there. As second option, one member of Crowd A (n_l) can upload his local messages to a server, which offers the possibility of a server synchronization with any member n_m of Crowd B; subsequently the message can further spread via peer synchronizations.

In order to satisfy the by design wanted confidentiality and unlinkability of messages, they are encrypted using a variant of the ElGamal encryption scheme that offers the possibility of re-randomizing ciphertexts without knowledge of the public key (for details see Section 3.2.2). The sender of a message can designate the message to a certain group by encrypting it with an appropriate group key; the exchange of group keys is based on social trust and outlined in Section 3.2.4. Whenever a peer receives messages, it checks whether he can decrypt them using any available group keys. By default messages get re-randomized before being sent to other peers during peer synchronizations.

In summary, a peer-based intra-crowd message dispersal is assumed, where the transmission of encrypted messages takes place in an opportunistic manner using a prioritized peer synchronization strategy. The inter-crowd dissemination with servers is included, since crowd segmentation may occur and the movement of nodes may not fully overcome these segmentations. Therefore, nodes can sporadically go online for server synchronizations (cf. Section 3.2.5).

System Overview

A schematic overview of the peer synchronization process is given in Figure 3.2: nodes belong to groups and each node has a group key for each of its *own groups*, i.e., all the groups it is a member of, in order to be able to encrypt and decrypt messages belonging to those groups. These keys are stored in a node's key store, e.g., the *Keys* of Node_1.

Each node maintains its *local storage* of encrypted messages, containing messages of groups it is a member of, as well as messages of *other groups* it is not a member of (and unable to decrypt). In this way, a decentralized storage of messages is created across all nodes. Any node can *encrypt* a new message under one of its group keys and put it in its storage for propagation.

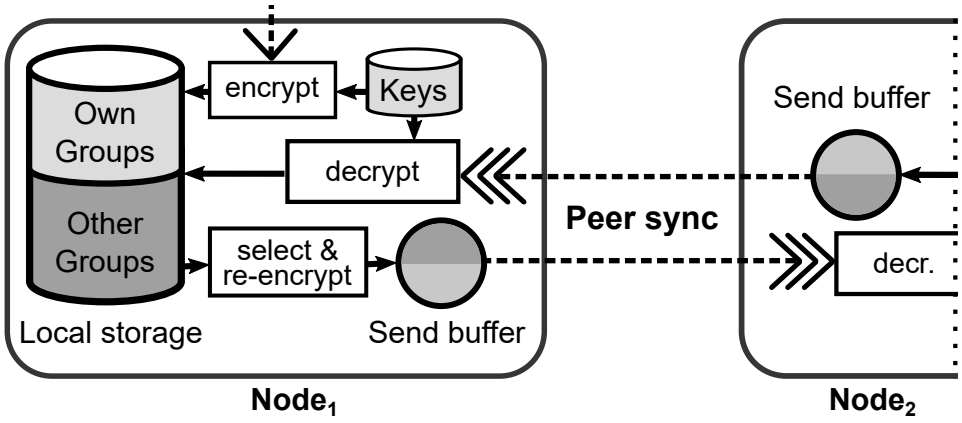


Figure 3.2: *Schematic overview of the peer synchronization process.*

Send buffers contain the messages to be exchanged in peer syncs. Prior to each peer sync they are filled in accordance to a fixed synchronization strategy. This determines how nodes *select* the send buffer messages out of their local storages. The different synchronization strategies used are described later in Section 3.2.5 of this chapter.

Peer synchronizations propagate the messages and are initiated whenever two nodes are in proximity of each other. During a peer synchronization two nodes, i.e., Node_1 and Node_2, bidirectionally exchange their *a priori* prepared send buffers. The exchange is conducted over a direct wireless point-to-point link established between the nodes *ad-hoc* when they are in communication range. Upon receiving a send buffer a node has to *decrypt* each message and check if it belongs to one of the groups it is a member of by trying to decrypt every message by every key it possesses. Then the messages are sorted into its local storage. In this way, messages are distributed across the nodes.

To achieve unlinkability – and thus anonymity – all nodes have to *re-encrypt* the encrypted messages each time before being resent in a peer synchronization. The cryptographic scheme must allow any node to re-encrypt any already encrypted message, regardless of whether the node is in possession of that specific group key or not. Therefore, a cryptographic scheme capable of universal re-encryption such as [56] has to be used.

Intuitively, the construction supports the following security properties: the goals of sender and receiver anonymity are achieved by the unlinkability of the in- and outgoing messages relayed from node to node via peer synchronization combined with the selection of messages pooled in the local storages. By encrypting all messages confidentiality is kept. Censorship-resistance is achieved by the decentralized, distributed nature of the peer-to-peer system which has no single point of failure.

The key differences of the approach used in this thesis to achieve anonymity compared to the approaches crowds and mixes (cf. Section 2.1), are the following: both crowds and mixes contain specific (uni-)cast routing and they are both developed for session-oriented fixed-line web communications, whereas our approach is based on a delay-tolerant networking scenario. This means that the messages spread agnostic to routing by direct communications only, i.e., peer synchronization. In this way the proposed solution has no routing overhead at all. Importantly, universal re-encryption is deployed instead of a layered one in this approach. This ensues, as stated in Section 3.2.2, that no message has to be decrypted multiple times in order to obtain the plaintext, albeit the node has to decrypt the message under the correct key by trial and error until the node has either no other keys to decrypt or has decrypted the ciphertext to the correct plaintext. This simplicity is by design in order to build a decentralized and peer-to-peer based microblogging infrastructure.

3.2.2 Universal Re-Encryptable ElGamal

A peer who is not a member of a group is not able to decrypt messages sent in that group. Also, a peer is not able to learn anything from messages he cannot decrypt (such as the public key it is encrypted with). Nevertheless he needs to be able to re-randomize all encryptions to provide unlinkability.

It is well-known that ElGamal ciphertexts can be re-randomized, but this operation requires knowledge of the public key. However, we can not employ such a solution directly, as it would contradict sender anonymity, since knowledge of the public key associated with a message allows to determine group membership of the message. In order to be able to re-randomize ciphertext even in case the public key is not available, a variant of the ElGamal encryption scheme introduced in [56] is used to obtain universal re-encryption.

In a cyclic, multiplicative group $(\mathbb{G}, \cdot)$ of prime order p with neutral element 1, on input of an ElGamal public key $pk = (g, h) \in \mathbb{G}^2$ and a message $m \in \mathbb{G}$, the encryption is composed of two parts: (i) an ElGamal encryption of m using pk and (ii) a random encryption of the neutral element 1, which allows for re-randomizing the ciphertext. Thus, the following tuple corresponds to an encryption of a message m, where two integers $r_1, r_2 \in \mathbb{Z}_p$ are chosen uniformly at random:

$$c = (c_1, c_2, c_3, c_4) := (g^{r_1}, h^{r_1} \cdot m, g^{r_2}, h^{r_2}).$$

Now (c_1, c_2) is a textbook ElGamal encryption of m while (c_3, c_4) is a random encryption of the neutral element 1. Re-randomization can be performed by choosing two integers $t_1, t_2 \in \mathbb{Z}_p$ uniformly at random and computing the re-randomized ciphertext c':

$$c' = (c'_1, c'_2, c'_3, c'_4) := (c_1 \cdot c_3^{t_1}, c_2 \cdot c_4^{t_1}, c_3^{t_2}, c_4^{t_2}).$$

Due to the homomorphic properties of ElGamal the tuple (c'_1, c'_2) is again an ElGamal encryption of some (possibly unknown) m and (c'_3, c'_4) is another encryption of 1 (both under the same public key). Re-randomization is based entirely on information present in the ciphertext and access to the public key pk is not required. For decryption, since the public key under which the encryption was done is unknown, it must first be checked if (c'_3, c'_4) decrypts to 1 before decrypting the actual message.

3.2.3 Message Format and Storage

The message format follows the hash-and-encrypt concept to achieve integrity, where a keyed hash is used (Message Authentication Code, HMAC) instead of a plain hash to achieve authenticity within the set of group members. Logically, a message m consists of a timestamp t concatenated with the 140 character message text msg preceded by their HMAC,

$$m = \mathrm{HMAC}(t \mid\mid msg) \mid\mid t \mid\mid msg.$$

Subsequently, the message is ElGamal encrypted as described in the previous section. (Note that any kind of hybrid encryption is not possible here, as it would not easily allow re-randomization of the message when it is forwarded; note further that due to the randomized nature of ElGamal the use of a special mode of operation is not necessary.)

Each peer maintains a message buffer of a fixed size, which is filled with incoming messages. Upon receipt of any new message, a peer checks whether it belongs to a group that the peer is interested in. This is done by brute-forcing: the peer tries to decrypt the message with all private group keys the node has; if decryption works, the message is part of a group that the peer is interested in. The decrypted message is shown to the user unless the message was already received before. This brute-force decryption step is necessary because we do not tag messages with any sort of group identifier, which would open the possibility of message linking attacks.

3.2.4 Group and Key Management

We assume that the global system parameters of the encryption scheme, i.e., the cyclic, multiplicative group $(\mathbb{G}, \cdot)$, is already present in the implementation of any client. Whenever a new message group is formed, the responsible peer – of which every group has one – creates an asymmetric ElGamal key pair, which identifies the new group. All members of a group possess both the public key and the private key of the ElGamal key plus a group-specific secret required to compute the HMAC for their group messages.

A key propagation mechanism that is based on social trust is assumed.[1] Whenever two nodes are close to each other, one node can "introduce" the other one to a group by initiating the exchange of the group ElGamal key pair. This key can, for example, be sent from one device to another one using NFC transmission or an optical channel (such as a barcode that is scanned by the other device). Key revocation is done by forming a new group and discarding the old group keys.

An important aspect for the microblogging system to work is thus the key sharing and propagation mechanism. Ideally, it should be intiuitive as well as secure. One approach to remedy this problem is devised in [83]. The idea is to use common hand shaking of the users with their small mobile devices in their hands – such as smartphones in the scenario treated in this thesis – to mutually authenticate users and subsequently exchange keys. The approach is based on sensing and analysing the actual shaking movement of two users' devices to securely authenticate the users. In a user study the method was proven to be intuitively usable. The interested reader can find a comprehensive survey of mobile device association techniques in [25]. Nevertheless, key management and sharing in this thesis are treated as a re-

[1] Note that this problem is not treated in this dissertation.

placeable black box, with the possibility of advanced group key agreement schemes being implemented at another point in time.

The use of group encryption schemes contained in an outer ElGamal encryption layer would be a possible extension. But adding or revoking users is costly and requires nodes to be online for re-keying [13]. A group in this scheme can also re-key or decide to use a key pair only for a specified period of time and then use another key pair; as long as the system-wide public parameters for the key generation are used, the newly created keys are usable out-of-the-box.

3.2.5 Synchronization of Messages

The architecture supports two synchronization methods, peer synchronization (in short peer sync) and server synchronization (in short server sync), which are described subsequently.

Peer Synchronization

Nodes in this scenario do not manage any routing information for peer sync operations, but use local communication (such as such as Bluetooth or ad-hoc WiFi connections) once other peers come into reach. Optical links such as infrared on the IrDA protocol stack are also feasible means of message transfer, but not commonly present in current smartphones [15].

The message buffers of all peers are of fixed size, and there can be different strategies to prioritize messages during peer syncs. We stress that this buffer does not only contain messages that the peer can decrypt, but also other messages; this is necessary to ensure an appropriate spread of all messages. The following strategies[2] are proposed:

Prioritized A fraction p of the send buffer is allocated to messages of those groups that the sending peer has subscribed; the other fraction $1 - p$ of slots goes to other messages. Both slots are filled randomly with respective messages from the message buffer. $\mathrm{PR}_{0.4}$ syncing thus means that 40% of the buffer is filled with messages from groups the node is a member of, and 60% are filled uniformly at random with messages of other groups. This allows to prioritize own messages.

Random The send buffer is filled uniformly at random with messages from the node's message buffer.

[2]Note that the strategy *Round Robin* and *Latest Only* have been dropped during the simulations, since the results of *Round Robin* have been similar to *Random* and the results of *Latest Only* did not show sufficient message spread in most scenarios.

Round Robin The send buffer is sequentially and block-wise filled with messages out of the local storage.

Latest Only Only the latest messages received are put into the send buffer and sent at the next sync operation.

The propagation speed of the messages is clearly dependent on the deployed synchronization strategy. The random drawing of messages is beneficial for privacy, as it does not signal group memberships to peers, in contrast to the prioritized mechanism.

In order to not leak the amount of exchanged messages, a fixed number of messages is sent (alternatively a lower bound on the number of sent messages can be established). To fulfill these constraints (in particular during the initial bootstrapping phase at the beginning of the use of the microblogging system), dummy messages are randomly created by a peer; alternatively a peer can engage in a first server sync to fill its message buffer before performing a peer sync.

The used technology for the point-to-point communication allows peers to initiate a peer sync manually, automatically, or semi-automatically. A manual peer sync would require peers to consciously connect their smartphones with other peers (e.g., by a short-ranged optical link). In manual mode the peer sync takes place only if it is actively confirmed by the user, based on existing social trust relationships. In automatic mode the peer sync runs in the background in discovery mode and syncs whenever another peer is available and in reach. In semi-automatic mode a user needs to be added to a white-list first before syncs can take place. For example, for direct wireless LAN connections this could be achieved by recording the Medium Access Control (MAC) address of the network adapter. This strategy lowers the probability of peer syncs with untrusted devices. Nodes are thus able to synchronize messages more or less restrictively, based on how risk-averse they are.

This means, there is a continuum ranging from very risk-averse behavior, i.e., only manual peer syncs with known devices in range takes place, over risk-neutral, i.e., semi-automatic peer syncs with automatic peer-syncing with known devices and manual syncing for unknown devices, to risk-prone, i.e., automatic peer syncs with all reachable devices regardless of whether they are known or unknown devices. The more risk-averse the node is, the less messages get relayed over it, diminishing the overall performance of the system. The more risk-prone a node is, the more messages get relayed over it, augmenting the overall performance of the system. However, nodes expose themselves more if they peer-sync semi-automatically or automatically, making them more susceptible for adversarial attacks.

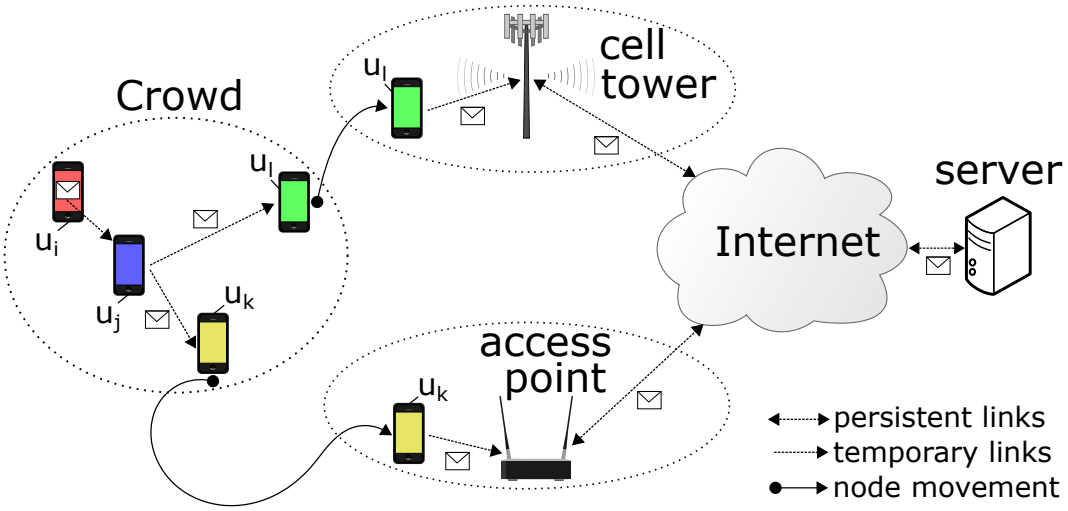


Figure 3.3: *Extended infrastructure overview of the microblogging architecture.*

Server Synchronization

Since different crowds of nodes are likely to get separated if they are not geographically close or socially connected, the option of downloading encrypted messages from servers is provided. Node movement might be a remedy against the segmentation of crowds over different boroughs of a city and even different towns. Nevertheless, if crowds are segmented by country borders or continents, a message spread only via node movement is improbable to unachievable. Server syncs thus provide an alternative means of message transportation in these scenarios. As seen in Figure 3.3, the connectivity might either be given through mobile telecommunication networks such as GSM (Global System for Mobile Communications), UMTS (Universal Mobile Telecommunications System) and LTE (Long-Term Evolution) cell towers, or wireless LAN access points.

During a server sync a node uploads its send buffer to the server and downloads new messages from it. Servers are only a data sink that can neither decrypt the messages, nor determine group memberships of messages. There may be several servers available for server syncs and they merely provide a means for on-demand message transfers (cf. Chapter 6 for both this basic exchange and the multi-client ORAM approach to server-sync messages).

A server sync is triggered by a node sporadically and only if it decides to do so. The Internet access could be done over an anonymity network such as Tor [39] to increase node anonymity during communication with a server.

3.3 Analysis

In this section we discuss to which extent the stated privacy and security goals can be met according to the assumed adversary capabilities stipulated in Section 3.1.4.

3.3.1 Privacy

The privacy of a node during any synchronization is guaranteed, since the exchanged messages look completely random to a global passive adversary and no group memberships can be derived from the encrypted messages themselves (due to the unlinkability and indistinguishability of the re-encrypted messages). Messages are peer-synced by peers irrespectively of the nodes' group memberships and the re-randomized messages can not be identified, linked or traced by the adversary. Only if a prioritized synchronization strategy is used that favors messages that a peer is a member of and if a malicious node compromised the specific key, group affiliations can be leaked. Generally however, the adversary does not know if a (non-compromised) node is able to decrypt any of the messages or not (thus learning its group memberships) because the internal processing of the received messages by the node is not visible to the adversary, e.g., the messages that the node is able to decrypt and read. Thus, the adversary is not able to draw information based on the observed ciphertext exchanges and it can not derive group affiliations, resulting in the confidentiality of group memberships and message content.

3.3.2 Anonymity

The solution's goal is to achieve full anonymity by the means of unlinkability, implying receiver and sender anonymity. Especially the peer synchronizations in the delay-tolerant networking are beneficial for anonymity, because the original sender of a message and any receivers can hardly be traced by the global passive adversary who is unable to decrypt the messages.

Server synchronizations require wide area network (WAN) communication using the smartphone. These means of communication are more susceptible to monitoring because data retention of cell towers and WiFi logins exists which allows for accurate tracking via device identifiers, e.g., International Mobile Subscriber Identity (IMSI), or International Mobile Station Equipment Identity (IMEI). Since server syncs are optional and queried by nodes on-demand only, the only information the adversary can gain is that the node is participating in the network and no derivation of original senders is possible.

3.3.3 Censorship-Resistance

The resistance to censorship is based on the distributed and decentralized storage of the messages, the local peer syncs, and a censor who has no access to the group keys. But even if the censor was able to get to know some keys, he is not able to fully stop the exchange of the messages encrypted under those keys. This is the case, since the censor only has a limited visibility of the P2P communications taking place and cannot shut down all the peer syncs that take place altogether (cf. P2P adversary in Section 3.1.4). Filtering on a semantic level is also not possible, neither by the passive global adversary nor by malicious nodes, because of the indistinguishability of all ciphertexts. Again, if the censor was able to get hold of some keys and made them available to the malicious nodes, they could filter out the messages encrypted under those keys, thus lowering – but not stopping – the spread of the affected messages. The result is therefore the inability of the censor to read or trace any message or to perform content-based filtering. This leaves a complete or partial communication shutdown or deletion of ciphertexts as possible measures for the adversary to censor.

For all WAN communications, disabling the server would suffice, as would regional communication shutdowns or disruptions of the Internet; both of which are not preventable. However, the architecture is designed to operate without WAN communications exclusively using local communications. Therefore, the architecture has to withstand local jamming of the network by an adversary that deploys jammers to block radio frequencies. This renders radio links in the jammer's sphere of influence unusable, because of electro-magnetic interference.

A potential remedy to jamming is the use of optical point-to-point links, although currently uncommon in smartphones. The line of sight required for optical peer-syncs is beneficial in this scenario and technologies such as *IrSimple* [2] and its successors would allow for bulk message synchronizations in less than one second. If optical peer-syncing is carried out based on social trust relations/personal encounters, censorship could be further hampered.

Consequently, the approach taken in the simulation runs with censorship (cf. Section 4.3.2) allows the censor to limited local jamming, so that a certain percentage of the peers in the network are blocked from conducting peer syncs.

3.4 Limitations

The task of building an anonymity system is not a simple undertaking. As outlined in [35], it is achievable if no one is logging or watching, or if the communication is of an one-off type, e.g., whistle blowing by shipping a flash drive once only requires the sender to engage in communications one time, and not repeatedly, thus minimizing the chances of being observed. Also, if infinite budget for covert traffic is feasible or if

steganographic or covert communications with high latency and a low bandwidth are sufficient, anonymity can be achieved rather easily. The same applies if using malware and bot nets is a viable option to create cover traffic. However, anonymity is a severe problem if the communication is periodic and persistent, and if the anonymity has to withstand against an adversary able to observe things, i.e., adversaries that have a high visibility of the network. Also using solely commodity of-the-shelf hardware and communication technologies makes building an anonymity system more difficult. The same applies to a high-bandwidth, low-latency network if a high reliability is needed.

The mobile anonymous microblogging system devised in this thesis has of course no easier standpoint when it comes to building an anonymity system. The smartphones used are equipped with a subscriber identification module (SIM) that is intended to securely store the international mobile subscriber identity (IMSI) number. The mobile network providers can thus trace the movement of the smartphone as it logs into the cell towers in communication range using this IMSI-number. Detailed profiling of movement patterns is thus possible. Combined with limited local monitoring and logging of communications this could, however, create a communication graph of who is performing peer syncs with whom, e.g., by technically collecting quasi-identifiers of the device such as media access control addresses (MAC-addresses) for WiFi. Even an adversary that has only a limited visibility of the P2P communication that takes place could then use community detection to reveal the communication patterns and show the degree of connectivity of a node and the frequency of its peer syncs. These results could be leveraged as a side-channel for assumed group affiliations. If the peer syncs were conducted by optical links over a short distance, they would become technically undetectable based on radio emissions. Local monitoring would then be difficult and require "shoulder surfing", i.e., surreptitiously scrutinizing someone else's smartphone activities over the user's shoulder. Also, correlation attacks of side-channel information from OSN social graphs with anonymous location patterns of users have been shown to successfully identify nodes [120]. These attacks can only be circumvented by users through either disabling communications that allow location tracking or by actively obfuscating their location data [116].

Similarly, Internet service providers (ISP) can log the Internet communication that takes place, often due to lawful interception and data retention requirements. This information is thus generally produced during normal usage of commodity of-the-shelf hardware. This has been stated in Section 3.3 and additionally the stipulated adversary model in Section 3.1.4 covers such malicious behavior to some extent. Yet, this information is generally gathered during normal usage and can only by mitigated by the use of an anonymity system such as Tor.

Nevertheless, in a real-world scenario compromised nodes are to be expected. Self-

ish nodes in the architecture might only propagate their own groups' messages and could thus compromise their group memberships in case of a compromised group member or group key.[3] Compromised key-material after the theft of a device or its infection with malware such as hardware trojans is possible. Such occurrences are not the exception but rather the rule and must be met by an anonymity system in order to mitigate possible damages. Backdoors, i.e., device functionality hidden from the user and only accessible to the developers which have implanted it on the device may also be triggered to compromise a node or update firmware without the user's consent. This is of course an ambivalent practice, since some users tend to not update their devices regularly and thus would have a longer window of vulnerability for known vulnerabilities; and the other backdoor vulnerabilites are intentionally persistent until they might get removed properly. On the other hand, users can also obtain malicious code on their devices via installed third-party software and any form of potentially unwanted programs. Also, trusted operating systems and a secured hardware and software stack cannot generally be anticipated in a real-world scenario which uses commodity of-the-shelf hardware and are out of scope for the considerations done in this thesis, leaving the nodes naturally vulnerable to a certain extent. This issue can only be circumvented by using dedicated devices, i.e., smartphones with deliberately minimized functionality, for instance solely for messaging or telephoning or emailing etc., in order to shield the target's surface. With respect to the problem at hand using dedicated hardened devices would inevitably shrink the userbase, an effect that contradicts the requirement stated in Section 3.2, i.e., the microblogging solution should work with commodity-of-the-shelf smartphones, and should be as intuitively to use as possible. As a result this means, even though the microblogging solution is designed in a way that the desired privacy and security guarantees are met, there exist factors during the actual deployment of the solution that are not controllable by the solution itself. The solution must then adapt to the threats as they arise and try to mitigate the possible damage of them.

Of course, there are also numerous other limitations besides the technical ones discussed above, e.g., socio-philosophical.[4]

[3] Please note that this is opposed to the wanted anonymity and privacy goals of the system.

[4] From a socio-philosophical perspective Privacy-by-Design(PbD) aims – in a continuum – to either negatively-defensively protect data or to positively-enablingly allow the user to autonomously define what personal data to share; with post-privacy advocates as one extreme. On the other hand, a culturally pessimistic view sees an era of *"Internet-Biedermeier"* with no *über*-individual content in a seemingly innate *"Universum der Gemütlichkeit"* (G. Anders) dawning. Are technologies a social *acteur*, defined ethno-graphically, cultural-anthropologically? Would a technocratic PbD thus be a paternalistic *"Gängelband"* allowing for no private communications in public communications? And mobile private communications thus only be a *"Kommunikation zwischen Fremden"* (R. Sennert) within the general public? [89]

3.5 Demonstrator

This section exemplifies how the microblogging solution could look like, when implemented on a smartphone – using peer syncs only. Section 3.5.1 illustrates the setup and the use cases of the demonstrator, and Section 3.5.2 shows the use cases of the actual prototypical implementation of the microblogging system.

3.5.1 Setup and Use Cases

This section shows the setup of the demonstrator and describes the use cases to be accomplished by the demonstrator. Note, that the demonstrator system uses only peer syncs.

The simulation server S runs a simulation of the mobile human users that move around and microblog as described in Chapter 4, i.e., the simulation creates all nodes, assigns their group memberships, creates their movements, and generates messages that get distributed via peer syncs. The users and their movements are visualized on the screen of the simulation server. Two real users A and B can use two actual smartphones as shown in the overall demonstrator setup in Figure 3.4. On the smartphones these users run a microblogging app which interacts with the simulation running on the server. The communication in this demonstrator is done using wifi; an access point connects the three entities. The basic use case is that the real users get automatically subscribed to groups, once they connect to the server. They can then obtain messages of the groups they have been subscribed to, or they can themselves create new groups. The joining of a new group can be done by a key exchange between the two users; it is realised by scanning QR-codes. For all the groups user A and B are a member of – initially assigned ones, newly created ones, or groups joined by physically obtaining the key from the other real user – they can create new messages on their own that get injected to the simulation, i.e., custom messages besides the ones auto-generated by the simulation. The injected messages

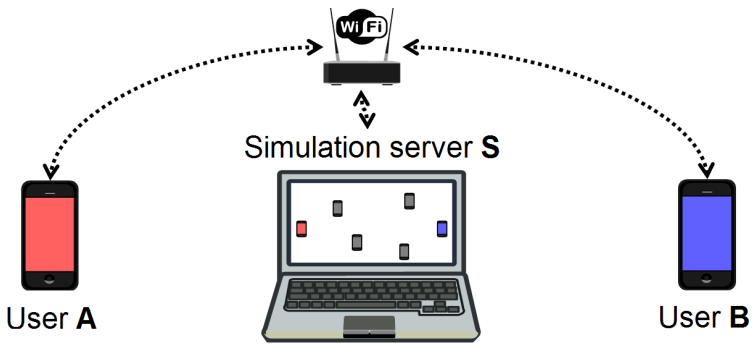


Figure 3.4: *Demonstrator setup showing the simulation server S and the two actual smartphones operated by user A and B, all of which are linked via an wifi access point.*

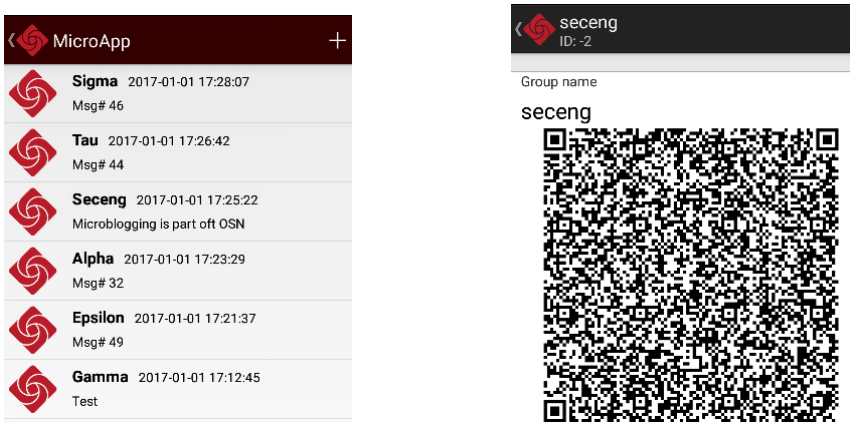
then become part of the simulation and they are spread via peer syncs as nodes have encounters. If both A and B are members of identical groups they can thus eventually receive messages from each other which they have themselves created and injected into the simulation.

3.5.2 Implementation

Based on the use case requirements an app was programmed for the smartphone operating system Android, that allowed two user-operated real smartphones to connect to the simulation. Figure 3.5(a) shows the main screen that contains a list of all the groups the user is a member of, followed by a timestamp of the latest message received in that group and the actual latest message received in that particular group. The messages that contain a text in the format *"Msg# x"* are auto-generated messages of the simulation, whereas all other messages are examples of user-generated messages that have been injected into the simulation. A user can touch upon a group entry and is then shown a screen with a timeline of all messages received in the selected group and a text input field that allows for the creation of new messages.

Group key exchange is done using QR-Codes, i.e., one user loads the QR-code represenation of the group key of which the key is to be exchanged to the screen – this is exemplary shown in Figure 3.5(b). The other user then uses the app to take a photograph of the shown QR-code group key and in this way receives the membership of this group. The app is designed in a way to allow for other mechanisms of group key exchange, e.g., Near Field Communication (NFC).

The use case of creating a new group can be carried out by the user by touching the settings icon of the application and then selecting the *Create new group*-button.



(a) Main screen. (b) QR-Code screen.

Figure 3.5: *Sample screenshots of the microblogging demonstrator app: (a) main screen showing the latest messages (b) screen for QR-Code-based group key exchange.*

The user is then prompted for a new group name in order to create the new group.

The mobile human users that move around and create messages have been made visible for the demonstrator by adding OpenStreetMap's map material and its waypoint information, i.e., a representation of the roads and ways that nodes can walk upon, to the simulation. The extension is realized as an Docker image that hosts an OpenStreetMap-Tileserver, containing a dump of the actual OpenStreetMap data of the project, alongside the waypoint information. Once an area of the map is selected by zooming into the area of interest, the simulation can be started and users get spawned on the waypoints and begin to move around. The users are restricted in their movements based on the selected map area, so that they cannot move out of the area shown on the screen. Whenever a node reaches a border of the selected area, it makes a u-turn in order to stay within the selected area. Figure 3.6 shows a snapshot of the simulation running with the OpenStreetMap-Tileserver and the users – depicted by the small cellular phone-like shapes – on the map's waypoints.

Figure 3.6: *Demonstrator: snapshot of the simulation with the OpenStreetMap-Tileserver and simulation-linked smartphone nodes on the map's waypoints.*

Chapter 4

Feasibility of the Message Distribution Using Peer Syncs

This chapter thoroughly evaluates and analyzes the feasibility of the mobile distributed microblogging system based on users carrying their mobile devices as described in Section 3.2 – using only peer syncs. As stated, the system goals are anonymity, message confidentiality, and censorship-resistance. All messages are encrypted under a group key, and they are stored across all nodes in a decentralized fashion.

The insights of this chapter are threefold:

- The feasibility of the mobile microblogging system is shown experimentally using simulations with both synthetic and empirical mobility datasets.
- Different networking setups are simulated to gain insights on how they affect message spread.
- The system is evaluated under adversarial conditions, its limitations are discussed, and lessons learned are stated.

The rest of the chapter is structured as follows: Section 4.1 details the different synthetic and empirical mobility datasets used to conduct the simulations, followed by a description of the simulation setup and the evaluation of the simulation's results in Section 4.2. The system's performance under adversarial conditions is investigated in Section 4.3. In Section 4.4 the findings are discussed and a conclusion is drawn.

4.1 Mobility Inputs

In order to evaluate the fitness for use of the presented solution, a discrete, event-based simulation is run which simulates the behavior of the smartphone carrying users, i.e., the nodes. It is the underlying mobility of the nodes that facilitates spreading of the messages. To that end, the movement of the nodes and its char-

acteristics are described by the means of networking metrics in this section.[1] Two different classes of mobility datasets are taken as input to the simulation: empirical data collected in three differing scenarios, and synthetic data created using the BonnMotion [3] mobility framework.

The following networking metrics are used to outline key characteristics of the data:

- Node degree is defined as the number of neighboring nodes which a node encounters within a period of two hours.
- Meeting duration in seconds is defined as the average duration of the encounters a node has.
- Meetings per node per day is the average number of meetings per day a node takes part in.

4.1.1 Empirical Mobility Data

Three empirical mobility datasets have been used in this thesis, from which input data to the simulation has been extracted:

- The Bluetooth dataset [111] comprises 285 smartphone users, i.e., nodes, communicating using the Bluetooth technology. The total duration of the dataset is 120 days and its source was an event at a congress center.
- The Nokia Mobile Data Challenge (NMDC) dataset [49] comprises GPS-traces of 186 nodes over a period of 576 days. The area covered was approximately 77,000km^2; however, the nodes were clustered in an area surrounding Lausanne. Yet, some wandered into France, therefore the large area.
- The SUVnet Traces dataset [132] comprises 30 days of mobility traces of 4445 taxis from Shanghai and its surroundings. The GPS traces cover an area of approximately 42,800km^2. The Wireless and Sensor Networks Lab of Shanghai Jiao Tong University provided these datasets.

Note that the NMDC and SUVnet datasets were originally GPS-traces and have been processed using the BonnMotion [3] mobility framework so as to calculate the earlier mentioned key characteristics of the datasets, e.g., node degree, meeting duration, meeting per node per day. The Bluetooth dataset was obtained in a data collection experiment where actual Bluetooth pairings between Bluetooth-enabled smartphones took place which were recorded.

[1] Please note that the empirical mobility inputs to the simulation were gained in a collaborative research project with significant involvement of A. Barroso. For a more thorough description of this data, please refer to [112].

For the purpose of the simulation, it was assumed that any two nodes are in communication range (and thus neighbors) if the distance between this pair of nodes was lower or equal to 15m. This distance is technology-dependent and was deliberately chosen, since it is at the discretion of us a reasonable upper bound if Bluetooth is used as communication technology.

The selection of the emprical datasets was done in such a way, that the chosen timeframe exhibits a high number of active nodes. This was necessary since the activity of the nodes varied in the datasets. Also, the datasets cover a timeframe of several months, a length that is considerably longer than the one needed for the purposes of the simulation. The empirical datasets obtained from the the raw input datasets are thus the following [112]:

- Congress1 and Congress2, obtained from the Bluetooth dataset: about 3.25 days each, 40 and 162 nodes, respectively.
- Lausanne1 and Lausanne2, obtained from the NMDC dataset: 2.5 days each, 141 and 137 nodes, respectively.
- Shanghai1 and Shanghai2, obtained from the SUVnet Traces: 2.5 days each, 4366 and 4347 nodes, respectively.

Table 4.1 shows the relevant statistics for the purpose of comparison. Besides node count, meeting count states the total number of encounters between pairs of nodes which took place. For node degree and meeting duration, also the mean, and standard deviation and median are listed, followed by values that indicate meetings per node, meetings per day and meetings per node per day.

We can see that the mean node degree for the Lausanne1 and Lausanne2 datasets (2.49 and 2.73) are about twice as high as the mean node degrees of the Congress1 and Congress2 datases, which are 1.19 and 1.39, respectively. The mean node degree of the Shanghai datasets is 17.05 for Shanghai1 and 15.56 for Shanghai2, approximately between five to sixteen times higher than the ones experienced in the Congress and Lausanne datasets.

The mean meeting duration in seconds of the Shanghai datasets are 15.5 and 17.0, and much lower than the ones from the Congress and Lausanne datasets which range from 420 to 7406. This is the case, since the Congress and Lausanne datasets are the traces of human beings at a conference and on campus, whereas the Shanghai traces are the ones of taxis.

The mean meetings per node per day show again values for the Congress and Lausanne datasets that are significantly lower than the ones for the Shanghai datasets. They range from 10.2 in Congress1 to 16.5 in Lausanne2 and have values of 130 and 145 in Shanghai2 and Shanghai1, respectively.

Table 4.1: *Connectivity statistics of the empirical dataset samples. Source: [112].*

	Congress1	**Congress2**	**Lausanne1**	**Lausanne2**	**Shanghai1**	**Shanghai2**
Node count	40	162	141	137	4366	4347
Meeting count	1332	7057	5826	5223	1585597	1412046
Total duration [s]	280885.0	281114.0	216000.0	216000.0	216000.0	216000.0
Node degree (ND) mean	1.19	1.39	2.49	2.73	17.05	15.56
ND standard deviation	1.82	3.17	8.47	7.68	18.43	17.46
ND median	0	0	0	0	11	9
Meeting duration (MD) mean [s]	420.1	3846.0	7406.0	6885.6	15.5	17.0
MD standard deviation [s]	1255.9	853.0	30495.9	21201.0	199.9	330.3
MD median [s]	150.0	150.0	234.1	254.3	2.9	2.8
Meetings/node mean	33.3	43.6	41.3	38.1	363.2	324.8
Meetings/day mean	409.7	2169.0	2330.4	2089.2	63423.8	564818.4
Meetings/node/day mean	10.2	13.4	16.5	15.2	145.3	129.9

4.1.2 Synthetic Mobility Data

In order to evaluate the simulation over a broader set of mobility input datasets, synthetic mobility models are used to generate mobility input datasets for the simulation. This is done to detect focal points at which the solution might not work anymore.

The ManhattanGrid [41] and the GaussMarkov [18] models supplied by the BonnMotion framework [3] were utilized to assess a wider set of mobility data. The ManhattanGrid model creates nodes that move on predefined paths laid out according to a regular grid structure, and the GaussMarkov model creates nodes that randomly walk and that change their direction of movement by chosing from a normal distribution whose mean is the respective old direction value. The node number has been varied from 25 to 400 in increments of 25 nodes per step, resulting in 16 different node numbers. The areas encompass a range from 1 to 70 km^2 in increments of 0.5 km^2 per step. The total duration of all synthetically created datasets is 195.300 seconds, which is approximately 54 hours or 2.25 days. The goal of these synthetic mobility datasets is to evaluate the boundary values of the densities in which the microblogging solution is working.

Selected connectivity statistics for a number of the ManhattanGrid and GaussMarkov datasets are shown in Table 4.2 and Table 4.3, respectively. Both tables show that the node degree is drastically decreasing once the area is larger than 10 km^2. In general, an augmented node number for a given area implies a larger node degree. This is particularly the case in areas from 1 to 5 km^2. The GaussMarkov datasets' average node degree lies in between 0.0002 and 0.28, while the ManhattanGrid datasets' mean node degree lies in between 0.0002 and 2. This is the case, since the nodes move on a fixed grid structure for the ManhattanGrid mobility and thus meet more often than the freely moving nodes on a plane in the GuassMarkov mobility.

The value of the mean meetings per node per day develops according to the node degree's pattern. The values spanned a range from 0.2 in the most sparsely connected ManhattanGrid dataset, i.e., 25 nodes on an area of 70 km^2, to 3210 in the most densely connected ManhattanGrid dataset, i.e., 400 nodes on an area of 1 km^2. For the GaussMarkov datasets, the mean meetings per node per day ranged from 0.3 in the most sparsely connected dataset to 578 in the most densely connected dataset.

A change of the area's size or a change of the number of nodes only exhibits an insignificant effect on the meeting durations. The GaussMarkov dataset-values spanned on average from 17.5 to 22.5 seconds, and the ManhattanGrid dataset-values from 17.6 to 29.5 seconds.

Table 4.2: *Connectivity statistics of selected ManhattanGrid datasets.*

Node count	Area km^2	Meeting count	Node degree (ND) mean	ND standard deviation	Meeting duration mean [s]	Meetings/node mean	Meetings/day mean	Meetings/node/day mean
25	1	10754	0.12	0.016	27.2	430.2	4779.6	191.2
25	5	1530	0.016	0.003	25.5	61.2	680	27.2
25	10	579	0.006	0.002	25.1	23.2	257.3	10.3
25	25	163	0.002	0.001	23.8	6.5	72.4	2.9
25	40	78	0.001	0	21.1	3.1	34.7	1.4
25	55	53	0	0	20.7	2.1	23.6	0.9
25	70	28	0	0	22.7	0.6	12.4	0.2
100	1	178384	0.503	0.027	27.5	1783.8	79281.8	792.8
100	5	24369	0.067	0.007	26.8	243.7	10830.7	108.3
100	10	9645	0.025	0.003	25.3	96.5	4286.7	42.9
100	25	2592	0.007	0.002	25.4	25.9	1152	11.5
100	40	1529	0.004	0.001	25.1	15.3	679.6	6.8
100	55	970	0.003	0.001	25.5	9.7	431.1	4.3
100	70	754	0.002	0.001	26.4	7.5	335.1	3.4
250	1	1125221	1.273	0.048	27.6	4500.9	500098.2	2000.4
250	5	152501	0.165	0.011	26.4	610	67778.2	271.1
250	10	60008	0.064	0.006	26	240	26670.2	106.7
250	25	17424	0.018	0.003	25.7	69.7	7744	31
250	40	9214	0.009	0.002	24.2	36.9	4095.1	16.4
250	55	6116	0.006	0.002	24	24.5	2718.2	10.9
250	70	4544	0.004	0.001	23	18.2	2019.6	8.1
400	1	2888818	2.05	0.06	27.7	7222	1283919.1	3209.8
400	5	388756	0.264	0.013	26.6	971.9	172780.4	432
400	10	154915	0.104	0.008	26.2	387.3	68851.1	172.1
400	25	44969	0.029	0.004	25.2	112.4	19986.2	50
400	40	24223	0.015	0.003	24.5	60.6	10765.8	26.9
400	55	16355	0.01	0.002	24	40.9	7268.9	18.2
400	70	11327	0.007	0.002	23.7	28.3	5034.2	12.6

Table 4.3: *Connectivity statistics of selected GaussMarkov datasets.*

Node count	Area km^2	Meeting count	Node degree (ND) mean	ND standard deviation	Meeting duration mean [s]	Meetings/node mean	Meetings/day mean	Meetings/node/day mean
25	1	1880	0.016	0.003	21.2	75.2	835.6	33.4
25	5	415	0.004	0.001	21.2	16.6	184.4	7.4
25	10	208	0.002	0.001	18.9	8.3	92.4	3.7
25	25	67	0.001	0	20.8	2.7	29.8	1.2
25	40	42	0	0	17.5	1.7	18.7	0.7
25	55	54	0	0	19.2	2.2	24	1
25	70	32	0	0	18.6	0.6	14.2	0.3
100	1	32046	0.068	0.004	20.7	320.5	14242.7	142.4
100	5	6676	0.014	0.003	20.8	66.8	2967.1	29.7
100	10	3249	0.007	0.002	20.5	32.5	1444	14.4
100	25	1313	0.003	0.001	20.7	13.1	583.6	5.8
100	40	782	0.002	0.001	20.9	7.8	347.6	3.5
100	55	554	0.001	0.001	20.1	5.5	246.2	2.5
100	70	477	0.001	0.001	21	4.8	212	2.1
250	1	202399	0.172	0.008	20.7	809.6	89955.1	359.8
250	5	41247	0.035	0.004	20.7	165	18332	73.3
250	10	20591	0.018	0.003	20.9	82.4	9151.6	36.6
250	25	8271	0.007	0.002	21.2	33.1	3676	14.7
250	40	4983	0.004	0.001	20.5	19.9	2214.7	8.9
250	55	3812	0.003	0.001	20.3	15.2	1694.2	6.8
250	70	2782	0.002	0.001	21	11.1	1236.4	4.9
400	1	520419	0.278	0.01	20.8	1301	231297.3	578.2
400	5	105195	0.056	0.004	20.9	263	46753.3	116.9
400	10	53369	0.028	0.003	20.7	133.4	23719.6	59.3
400	25	20945	0.011	0.002	20.7	52.4	9308.9	23.3
400	40	13268	0.007	0.002	20.7	33.2	5896.9	14.7
400	55	9638	0.005	0.001	20.7	24.1	4283.6	10.7
400	70	7649	0.004	0.001	20.5	19.1	3399.6	8.5

4.2 Simulation of Microblogging System

The mobility input determines the possible peer-sync events and constitutes the network on which the simulated microblogging system runs. The goals of the simulations are to gain insights on the message propagation using real empirical data as well as synthetic data. We first characterize the simulation runs in Section 4.2.1 and then we present the results of the simulations in Section 4.2.2.

4.2.1 Simulation Overview and Pseudocode

The message propagation of the system is assessed with a discrete, event-based simulation in which nodes synchronize their messages amongst themselves using local point-to-point communication links established when close to each other, i.e., peer syncs.

Flowchart Overview of the Simulation

The flowchart in Figure 4.1 provides a simplified overview of the simulation based on which we describe it in a more general way, before we give a more technical pseudocode description of the simulation. For the assessment of the message propagation

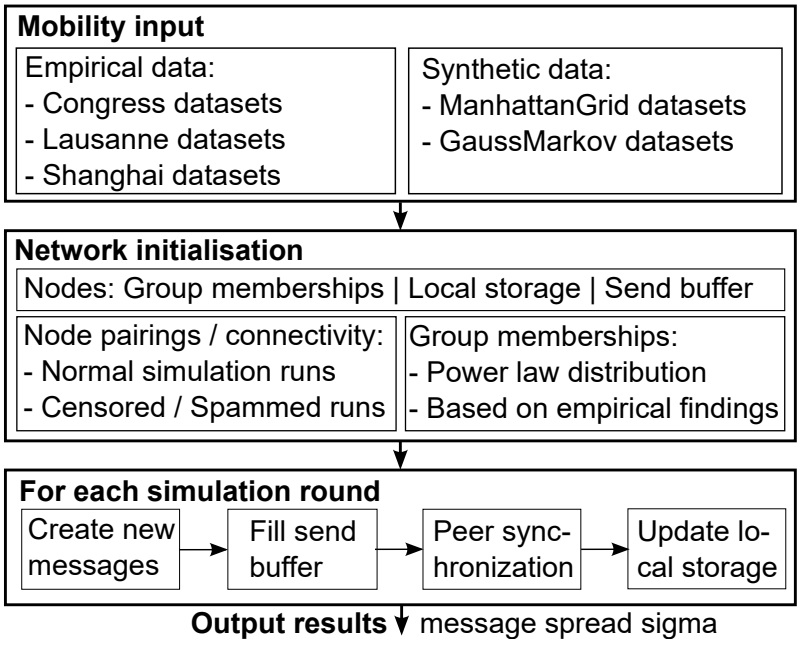


Figure 4.1: *Overview of the simulation.*

in the microblogging system, the implementation of the discrete, event-based simulation is conducted *excluding* central servers. The nodes, their movements and peer syncs are the major factors that influence the message propagation.

Mobility input to the simulation are the empirical and synthetic mobility datasets presented in the preceding section. They determine the nodes' movement on a plane according to the empirical or the generated mobility traces.

During *network initialisation* the nodes are created and their local message storage and send buffer is set. The group memberships of each node are also assigned at this stage. The formation of the groups are modeled to be similar to the empirical findings extracted from existing microblogging data. The group memberships of each node are drawn out of a discrete power-law distribution [28] with exponent $\alpha = 2.276$ and $x_{min} = 2$. The same applies to the number of groups a node is a member of – values which have empirically been computed in [74]. During simulations the group memberships remain fixed once they got initialized, i.e., there is no node churn. However, in reality the group assignment is done by peers spreading the keys among each other. The simulation results do not incorporate network churn, meaning that nodes are online and do not go offline during a simulation run. Not modeled due to a lack of accessible empiric mobility datasets are homophily and reciprocity, i.e., nodes or clusters of nodes that have frequent physical contact based on their mobility are assumed to be similar in their group memberships (interests). These correlations may have numerous consequences which are out of scope for this thesis [74].

The mobility patterns processed during the mobility input are the foundation used to derive node connectivity, i.e., the node pairings (peer syncs) that can take place. We determine whether a given pair of nodes is eligible for a peer sync by setting the maximum distance over which a peer sync can take place to 15 meters based on Bluetooth characteristics (the Bluetooth standard states connections over up to 100 meters, but tests with several up-to-date smartphone models showed it did not perform well for distances over 30 meters). For peer syncs, we do not take into account meeting durations. A peer sync can thus also occur in cases where the contact duration would be too short for current technologies to establish a link, e.g., the vehicular movements in the Shanghai datasets. (Note that this is not the case for the Congress datasets, in which all links established are based on actual Bluetooth pairings with successful data transfers that took place during data collection.)

Simulation rounds of 5 minutes duration are used in the simulation. At the beginning of each round, the nodes create their new messages. The message creation events, i.e., when a new message is initially created by a user, are sampled based on empirical Twitter microblogging user behavior [130]. During every round each group's number of newly created messages is obtained by a drawing out of a Poisson distribution with $\lambda_{Group} = 0.21 \cdot |Group|$, $|Group|$ being the number of peers in that group. The sampled number of messages for each round and group are distributed uniformly at random across the group members. Then, each node fills its send buffer with a specified number of messages drawn out of the node's local storage according

to the used synchornization strategy, e.g., Prioritized or Random. The third step consists of the peer synchronizations. Each peer sync makes a pair of nodes exchange their prepared send buffer as drawn by the fixed synchronization strategy. Finally, each node updates its local storage. The latest incoming messages from a peer sync thereby shift out the oldest messages received.

Output results of the simulation are the `sigma`-values as described in Section 4.2.2.

Pseudocode Representation of the Simulation

A more technical pseudocode overview of the simulation is provided in the listing below:

```
1  //1. Input:
2  // a. Mobility parameters
3  Mobility m = {"Congress1","Congress2","Lausanne1",
4                "Lausanne2","Shanghai1","Shanghai2",
5                "GaussMarkov1km2",...,"GaussMarkov70km2"
6                "ManhattanGrid1km2",...,"ManhattanGrid70km2"}
7  int iNum = 300
8  int rNum
9  int nNum
10 // b. Transmission parameters
11 int pMax = 3
12 bool isCensor = false
13 float pCensor = [0.00, ..., 1.00]
14 bool isSpam = false
15 float pSpam = [0.00, ..., 1.00]
16 int d = 15
17 // c. Node parameters
18 int bfSize = 100
19 int dbSize = 10000
20 Sync s = {"Prioritized", "Random", "Round Robin", "LatestOnly", "Psi"}
21 float pPrioritized = [0.00,...,1.00]
22 //2. Simulation:
23 // a. Create nodes and network
24 Node nodes[] = createNodes(nNum, bfSize, dbSize, s, pPrioritized)
25      nodes[].createGroupsAndMemberships(nNum)
26      nodes[].createAllMessageEvents()
27      if (nodes.s = "Psi") nodes.calculatePsiSetsForNodes()
28 Network net = createNetwork(m, nodes[], d, pMax, iNum, rNum, nNum,
29                              isSpam, isCensor, pSpam, pCensor)
```

```
30 // b. Run discrete round based simulation
31 for (int r = 0, r < rNum, r++) {
32    if(isSpam) net.setSpammingNodes()
33    if(isCensor) net.setCensoredNodes()
34    nodes[].updateSendBuffers()
35    nodes[].syncMessages()
36 }
37 //3. Output:
38 MessagePropagation sigma = [sigma_1, ..., sigma_rNum]
```

The input phase of the simulation, which is written in the object-oriented programming language Java, consists firstly of the `Mobility parameters`, namely the empirical mobility datasets and the synthetic mobility datasets `Mobility m` as described in Section 4.1.2 and Section 4.1.1, respectively. They determine when nodes are eligible for peer syncs. The variable `iNum` denotes the interval length of a simulation round in seconds, i.e., 5 minutes. The total number of rounds in the simulation `rNum` depends on the input `m` and in this case equals 500 rounds, i.e., around 1.7 days. `nNum` is the number of nodes, as described in Sections 4.1.1 and 4.1.2.

Further inputs are the `Transmission parameters`: `pMax`, which determines the maximally allowed number of peer syncs per node per simulation round, the Boolean value `isCensor`, which determines whether it is a censored simulation run or not (with the variable `pCensor` as the percentage of the nodes that are censored and can thus not peer-sync in a given round), the Boolean value `isSpam`, which determines whether it is a spammed simulation run or not (with the variable `pSpam` as the percentage of the nodes that spam), the variable `d`, which is the maximum distance in meters over which nodes can conduct peer syncs, and the variable `pPrioritized`, which indicates the percentage of a node's send buffer that is reserved for its own group messages in a peer sync.

In `1.c.`, the `Node parameters` are `bfSize` as the amount of messages transferred in a peer sync, `dbSize` as the maximum amount of messages stored locally in a node's database, and `Sync s` as the followed synchronization strategy. The standard buffer size is set to 100 messages and the local storage consists of 10,000 messages in each node's database.

The second phase, the begin ot the actual simulation, starts on line 22 and creates and initializes the objects `Node nodes[]` and `Network net` with the already described parameters in its first step `2.a.` Note that this creation takes place anew in each complete run of the simulation to have enough sample runs. For the nodes the functions `createGroupsAndMemberships(nNum)` and `createAllMessageEvents()` create the number of groups and their members and the message creation per group and node according to a power-law distribution and a Poisson distribution, respectively.

The function `calculatePsiSetsForNodes()` calculates the intersection of the group memberships for each pairwise different pair of nodes, if the synchronization strategy is `"Psi"` (cf. Chapter 5.1 in which this sync strategy is motivated and examined).

Step `2.b.` conducts the actual simulation rounds and is iterated for the number of rounds `rNum`. In each round first a check is done, whether it is a spammed simulation run or a censored simulation run, and the functions of the `net`-object `setSpammingNodes()` or `setCensoredNodes()` are used to mark its nodes as spammers or being unable to communicate, i.e., censored, respectively. The function `nodes.updateSendBuffers()` prepares the nodes' send buffers according to the chosen sync strategy `s`, e.g., Prioritized or Random. During this process a node first checks if it has created messages itself, and, if this is the case, it adds them to the send buffer. Then, the remaining send buffer is filled with messages of the nodes message database in accordance with the deployed sync strategy. The `nodes.syncMessages()`-function performs the actual syncs of the messages: all pairs of nodes that are within the maximum allowed distance `d` over which a peer sync can take place are eligible to conduct a peer sync. If a node has more possibilities to peer sync than the allowed maximum number of peer syncs per node `p`, the peer syncs exceeding `p` for this node are discarded. Each peer sync results in the exchange of the prepared send buffer between the nodes, followed by the adding of the received messages in the nodes' local databases.

In Section 4.2.2 we describe the `sigma`-values, which are the output of the simulation.

4.2.2 Simulation Results

Before we discuss the results, we first define global message spread σ and introduce a notation for parameterized link settings. The results are then first presented and analyzed for the empirical Congress, Lausanne and Shanghai mobility inputs, and then for the synthetic ManhattanGrid and GaussMarkov ones.

Global message spread σ

To evaluate the effectiveness of the microblogging system, we use the global message spread σ as metric, which is defined as

$$\sigma = \frac{\sum_{m \in M} \frac{rec_m}{|group_m|}}{|M|},$$

where M is the set of messages whose spreading is monitored, rec_m is the number of group members (including the sender) who received message m, and $|group_m|$ is the number of group members belonging to the group under which message m has been encrypted. Consequently, if one message m is received by all its group members,

its spread is 1 (full spread), and if all messages are fully spread, the global message spread is 1.

Parameterized Link Settings Λ_p^b

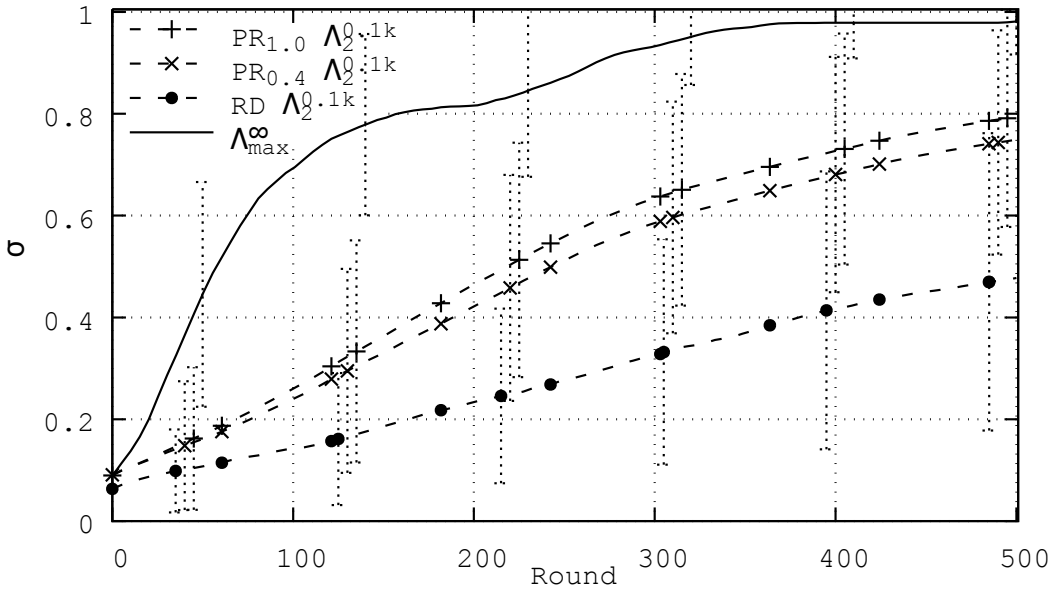
As shorthand notation for parameterized link settings we introduce Λ_p^b, with p being the maximum allowed number of peer syncs per round and node, and b being the size of the send buffer exchanged per peer sync. Various link settings are used in the simulations to assess their effects on message spread and to identify bottlenecks. *Bluetooth* ($\Lambda_2^{0.1k}$) link settings limit the number of peer syncs to 2, i.e., even if a larger number of peer syncs was possible a node peer-syncs with mostly 2 neighbors per simulation round, and the message send buffer to 100 messages, in order to adjust the link capacity settings to a contemporary transmission technology used widely. *Unlimited* ($\Lambda_{\max}^{\infty}$) link settings allow all possible links to be used for peer syncs, and the send buffer and the nodes' storages are unbounded. The Unlimited link settings are chosen to test a hypothetical upper bound of a possible message spread under conditions unconstrained by a given transmission technology and local storage. Note that the local storage is only unbounded for $\Lambda_{\max}^{\infty}$, otherwise it is set to 10,000 messages as stated above. Also, the peer sync strategies Random and PR do not have any effect for $\Lambda_{\max}^{\infty}$ peer sync results.

Empirical Mobility Data Results

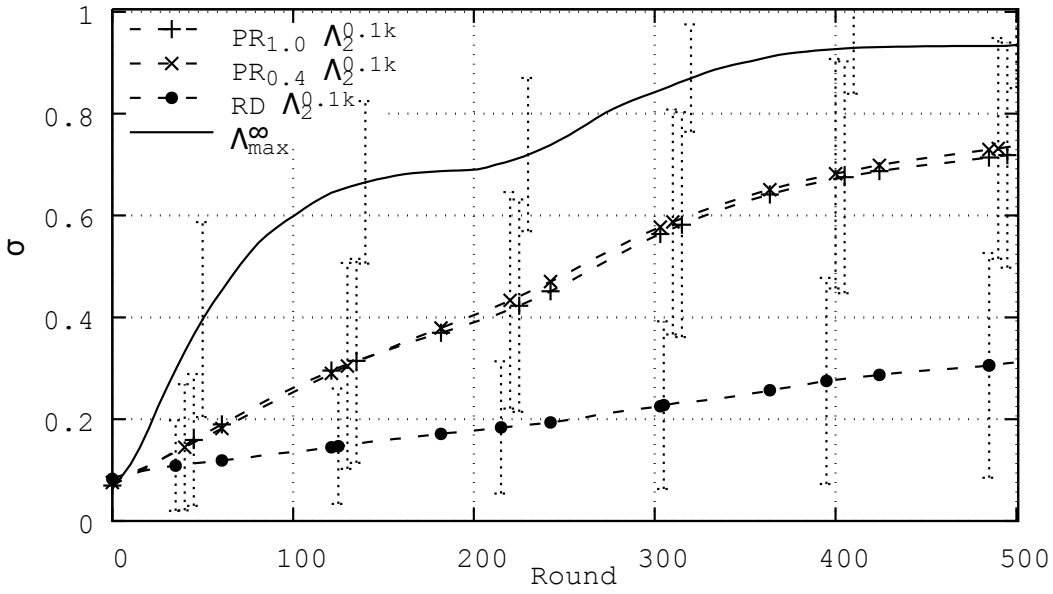
The results of the Congress1 (C1) and the Congress2 (C2) datasets are depicted in Figures 4.2(a),(b). They show the message spread over time for three different sync strategies with error bars for selected points in time using Bluetooth ($\Lambda_2^{0.1k}$) link settings and the upper bound obtained when unlimited ($\Lambda_{\max}^{\infty}$) link settings are applied. The smaller C1 sample reaches close to 50% global message spread after 500 rounds, the larger C2 sample slightly above 30% for random syncs. With $\mathrm{PR}_{0.4}$ syncs 75% is reached for both datasets, because the prioritized selection of messages yields a better message spread. Even though the standard deviation is large, PR peer syncs run with Bluetooth settings converge to the unlimited upper bound by a margin of approx. 15-20%, whereas RD peer syncs perform much worse.

The Lausanne1 and Lausanne2 datasets depicted in Figures 4.3(a),(b) show similar results: with Bluetooth settings, the Random, $\mathrm{PR}_{0.4}$ and $\mathrm{PR}_{1.0}$ sync methods yield final global message spreads slightly below 10%, and the spread with unlimited settings reaches around 55%, even though the node number in Lausanne is similar to the C2 dataset. These results are due to the large area covered by the nodes and the bipartitioned network structure whose two node clusters have little exchanges of messages between them as outlined in [112].

For the Shanghai datasets seen in Figures 4.4(a),(b) the results also show spreads

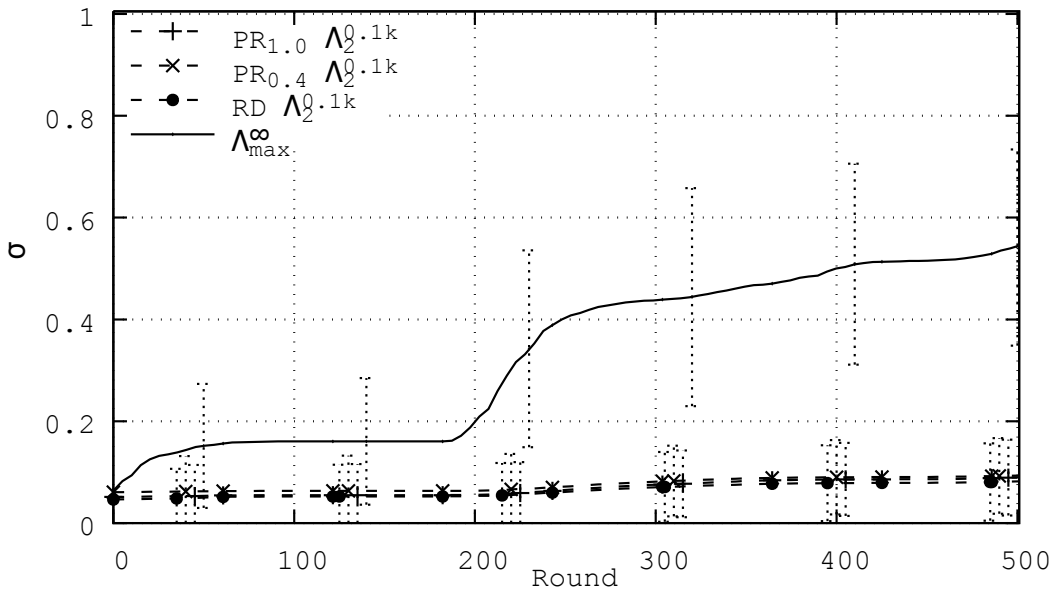


(a) Congress1.

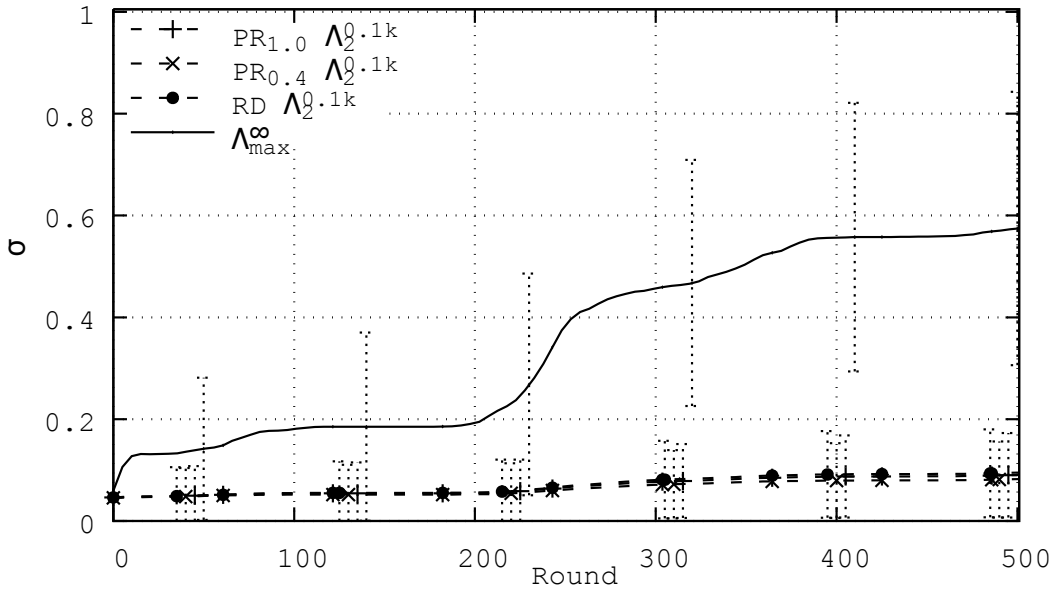


(b) Congress2.

Figure 4.2: *Congress empirical mobility: global message spread σ averaged over five runs with Bluetooth link settings for Prioritized ($PR_{0.4}$ $\Lambda_2^{0.1k}$, $PR_{1.0}$ $\Lambda_2^{0.1k}$) and Random (RD $\Lambda_2^{0.1k}$) peer syncs, and unlimited link settings for $\Lambda_{\max}^{\infty}$ peer syncs.*

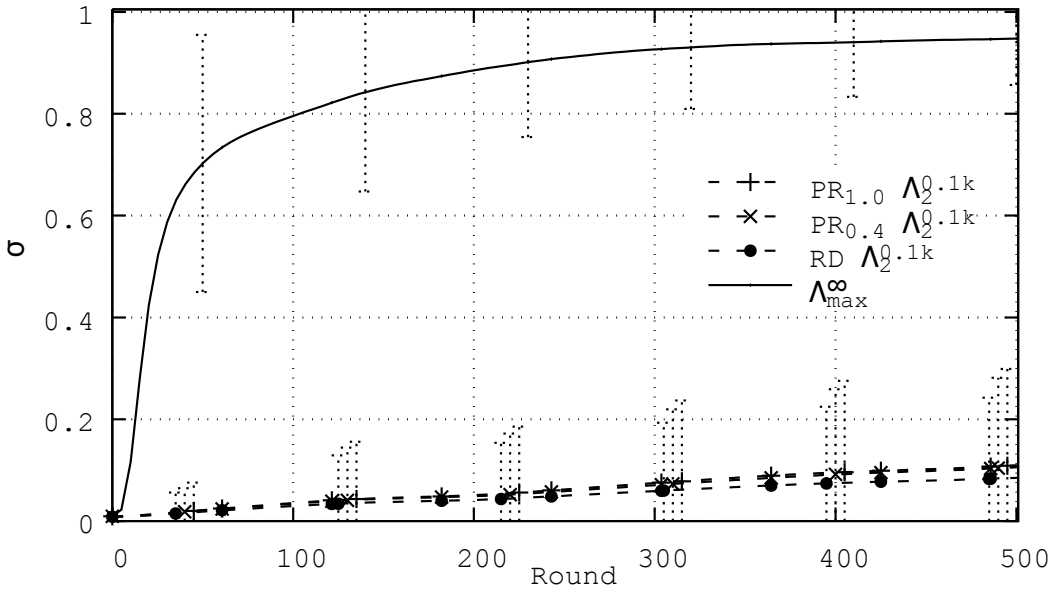


(a) Lausanne1.

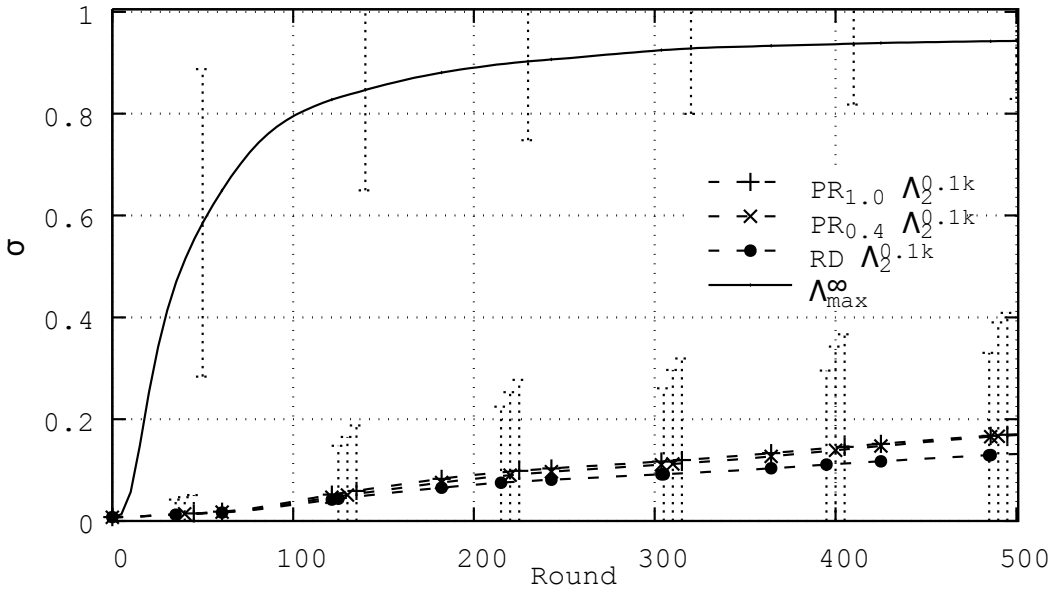


(b) Lausanne2.

Figure 4.3: *Lausanne empirical mobility: global message spread σ averaged over five runs with Bluetooth link settings for Prioritized ($PR_{0.4}$ $\Lambda_2^{0.1k}$, $PR_{1.0}$ $\Lambda_2^{0.1k}$) and Random (RD $\Lambda_2^{0.1k}$) peer syncs, and unlimited link settings for $\Lambda_{\max}^{\infty}$ peer syncs.*



(a) Shanghai1.



(b) Shanghai2.

Figure 4.4: *Shanghai empirical mobility: global message spread σ averaged over five runs with Bluetooth link settings for Prioritized ($PR_{0.4}$ $\Lambda_2^{0.1k}$, $PR_{1.0}$ $\Lambda_2^{0.1k}$) and Random (RD $\Lambda_2^{0.1k}$) peer syncs, and unlimited link settings for $\Lambda_{\max}^{\infty}$ peer syncs.*

between 10%-18% for peer syncs with Bluetooth settings and are thus similar to the results obtained from the Lausanne datasets, despite the fact that over 4000 nodes participate in the network. The results with unlimited $\Lambda^{\infty}_{\max}$ link settings reach 95% final spread, with a steep increase in the first 50 rounds to 70% for Shanghai1 and 60% for Shanghai2. This is caused by the high connectivity of the Shanghai datasets which exhibit a number of meetings per node that is about 9-10 times larger than the ones of the Congress and Lausanne datasets.

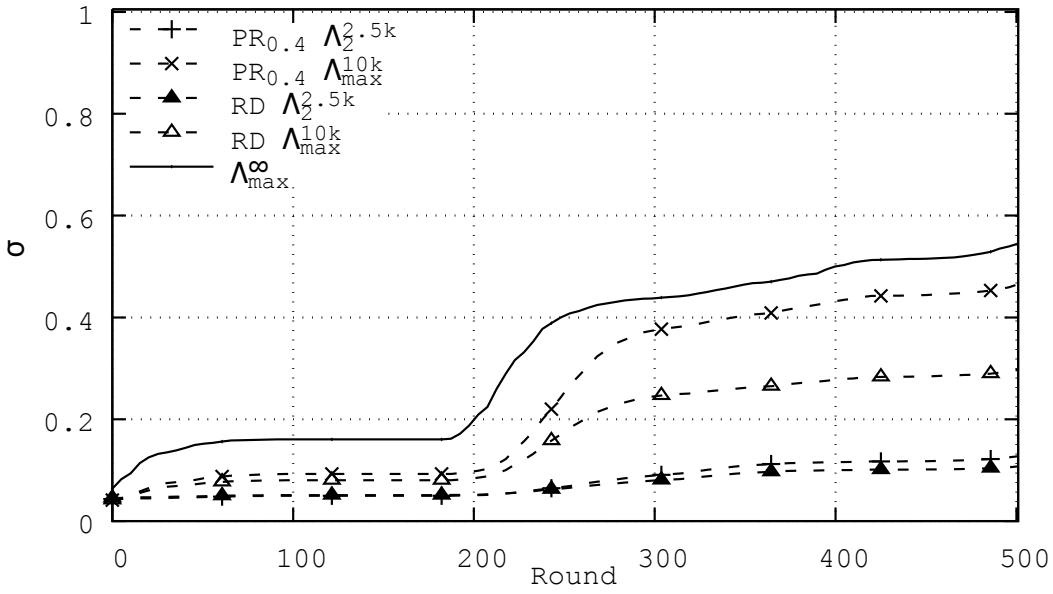
To approximate how well link settings other than the introduced Bluetooth setting can close up to the Unlimited setting, the hypothetical upper bound, a number of varied link settings have been run for the Lausanne and the Shanghai datasets.

The Figures 4.5(a),(b) show the results for the Lausanne1 and Lausanne2 datasets with varied link settings in comparison to the unlimited link settings. For the $\Lambda^{2.5k}_{2}$ link setting the final message spread in Figure 4.5(a) for Lausanne1 is only slightly better with approximately 11% for RD and 13% for $\mathrm{PR}_{0.4}$ than that of the Bluetooth $\Lambda^{0.1k}_{2}$ results shown for Lausanne1 in Figure 4.3(a). Contrarily, Lausanne2 shown in Figure 4.5(b) exhibits a significant improvement of the final message spread with the $\Lambda^{2.5k}_{2}$ link setting compared to that of the Bluetooth $\Lambda^{0.1k}_{2}$ results in Figure 4.3(b): it is now increased to approximately 18% for RD and 22% for $\mathrm{PR}_{0.4}$. Notably, the two peer sync mechanisms RD and $\mathrm{PR}_{0.4}$ do not differ much at the $\Lambda^{2.5k}_{2}$ link setting.

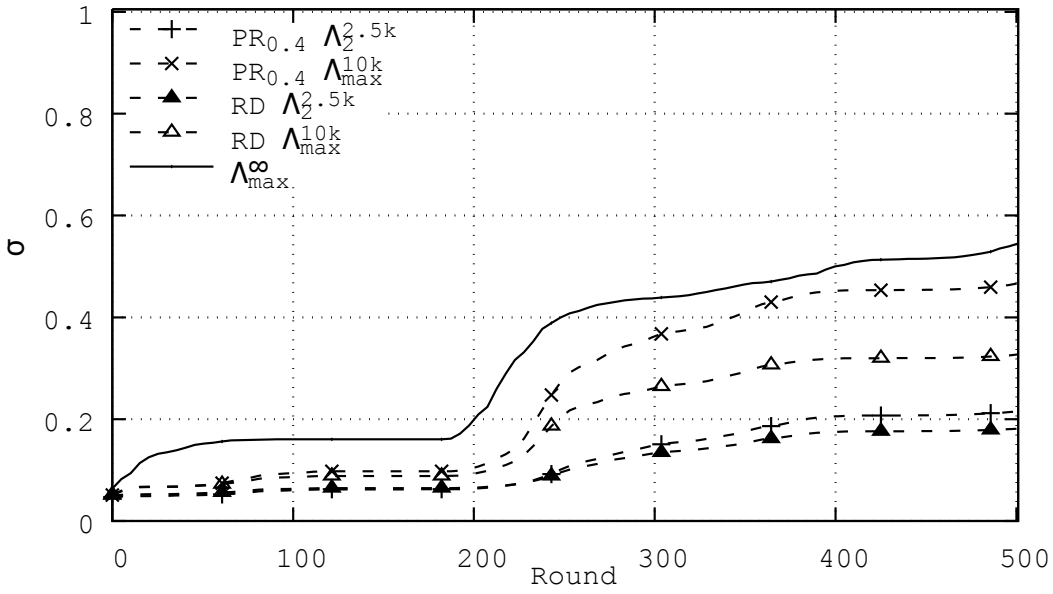
With a link setting adjusted to $\Lambda^{10k}_{\max}$ the final message spread with RD peer syncs reaches around 30% in Lausanne1 and approximately 33% in Lausanne2. The $\mathrm{PR}_{0.4}$ peer syncs with $\Lambda^{10k}_{\max}$ show a message spread that is closing up to the unlimited $\Lambda^{\infty}_{\max}$ link setting and which is significantly better than the RD syncs of the same link setting: for Lausanne1 it reaches around 46% and is only 8% lower than the unlimited link setting. For Lausanne2 it closes at approximately 47% and is about 10% lower that the unlimited link setting result.

The Figures 4.6(a),(b) show the Shanghai1 and Shanghai2 message spread over time with varied link settings compared to the unlimited link setting. With a link setting adjusted to $\Lambda^{2.5k}_{2}$ and RD peer syncs the Shanghai1 final message spread shown in Figure 4.6(a) is at 31%, and the $\Lambda^{2.5k}_{2}$ RD syncs final message spread for Shanghai2 shown in Figure 4.6(b) is at approximately 35%. Compared to the Bluetooth results of Shanghai1 and Shanghai2 depicted in Figures 4.4(a),(b) this is a significant improvement of message spread. Increasing the link setting to $\Lambda^{10k}_{\max}$ does not yield large improvements of message spread for RD peer syncs: for Shanghai1 the increase in message spread is only about 4%, and for Shanghai2 the increase is approximately 3% compared to the $\Lambda^{2.5k}_{2}$ link setting.

With $\mathrm{PR}_{0.4}$ peer syncs and a $\Lambda^{2.5k}_{2}$ link setting, the message spread is again raising and it is in the final round at approximately 44% for the Shanghai1 dataset and at approximately 53% for the Shanghai2 dataset. Adjusting the link setting to $\Lambda^{10k}_{\max}$

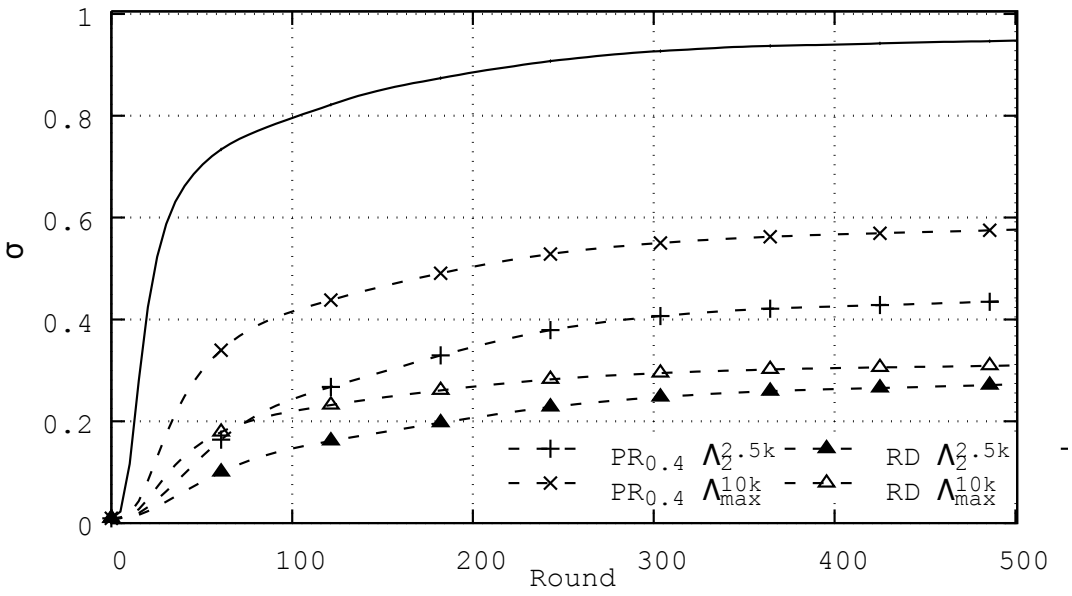


(a) Lausanne1.

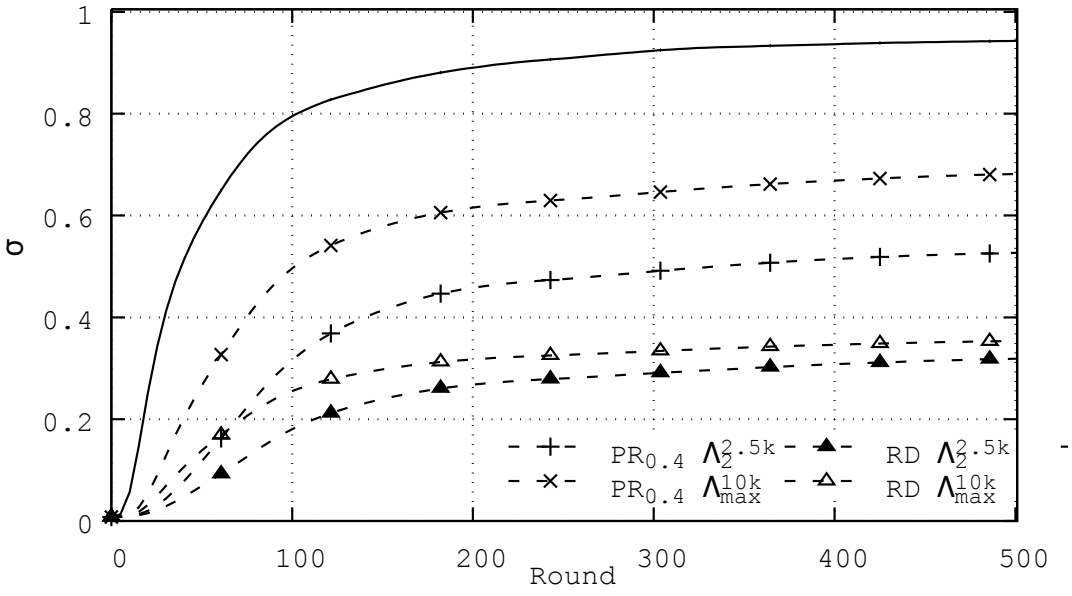


(b) Lausanne2.

Figure 4.5: *Lausanne empirical mobility: global message spread σ averaged over five runs with varied link settings for Prioritized (PR) and Random (RD) peer syncs and unlimited $\Lambda^{\infty}_{\max}$ link settings.*



(a) Shanghai1.



(b) Shanghai2.

Figure 4.6: *Shanghai empirical mobility: global message spread σ averaged over five runs with varied link settings for Prioritized (PR) and Random (RD) peer syncs and unlimited $\Lambda^{\infty}_{\max}$ link settings.*

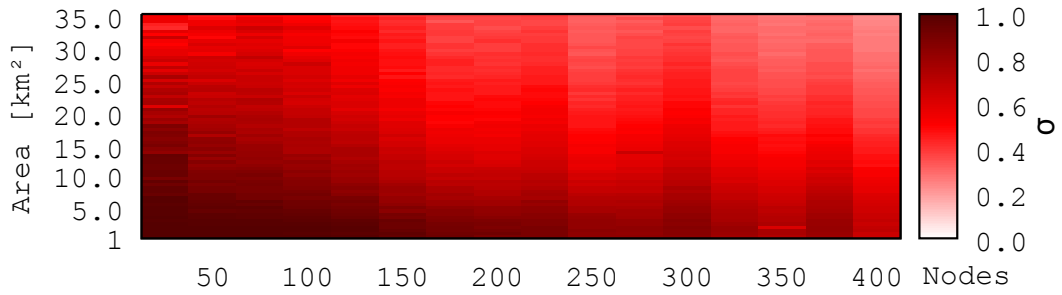
yields – just as for the Lausanne datasets – a larger increase of the message spread. The final message spread is advancing to around 58% for Shanghai1 and to approximately 68% for Shanghai2. Opposed to the Lausanne datasets the final message spread in the Shanghai datasets is not close to the hypothetical upper limit of the $\Lambda^{\infty}_{\max}$ link setting which is at about 95% for both of the Shanghai datasets. This is the case, since the number of nodes in the Shanghai datasets amounts to more than 4300 nodes, approximately 31 times higher than that of the Lausanne datasets, and the resulting message load on the network is also correspondingly higher. Apparently, the throughput at the nodes is not managing that message load that well as to facilitate higher message spread.

Concluding, it can be stated that the mobile microblogging using solely local point-to-point links works well, i.e., it achieves a 75% final message spread for sufficiently connected scenarios such as Congress1 and Congress2 with link settings adjusted to Bluetooth. In the Lausanne datasets the bipartition of the network prohibits a final message spread above roughly 50%, even for the unlimited link setting, and in the Shanghai datasets the transmission technology's bandwidth is the limiting factor, resulting only in fair final message spreads around 33% for higher-bandwidth Random syncs and acceptable final message spreads around 60% for higher-bandwidth Prioritized$_{0.4}$ syncs. However, with a higher bandwidth reasonable message spreads can also be reached for the Lausanne and Shanghai datasets. Yet, the Shanghai cases show that even with augmented link settings the completely decentralized microblogging using only peer syncs does not scale well. This shows that the system works for small node numbers, and that it could also be deployed for larger ones if communication technologies with a higher bandwidth were available.

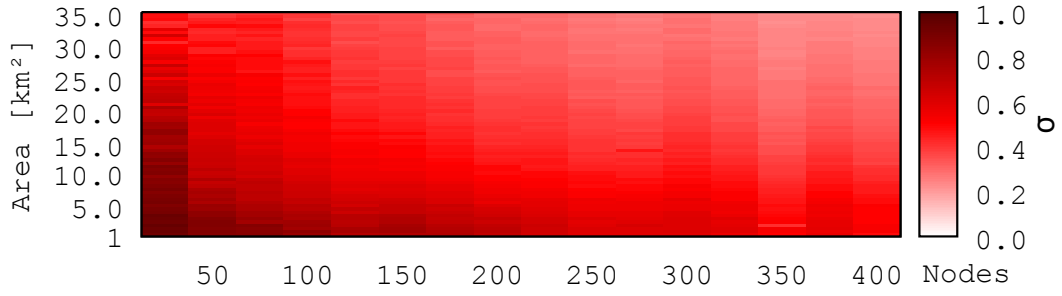
To gain a broader understanding of the ways messages spread, there is the need to conduct the simulations with synthetic movement patterns. This is the case, because the empirical movement patterns exhibit the characteristics of specific homogeneous groups, i.e., congress visitors, students and taxi drivers. These datasets could exhibit a movement and connection pattern different from that of datasets with heterogeneous users of cities. Therefore, to get more generalizable results, synthetic mobility datasets which cover a wider range of densities, need to be studied as well. This is particularly essential in order to discover focal points at which the microblogging system might stop working.

Synthetic Mobility Data Results

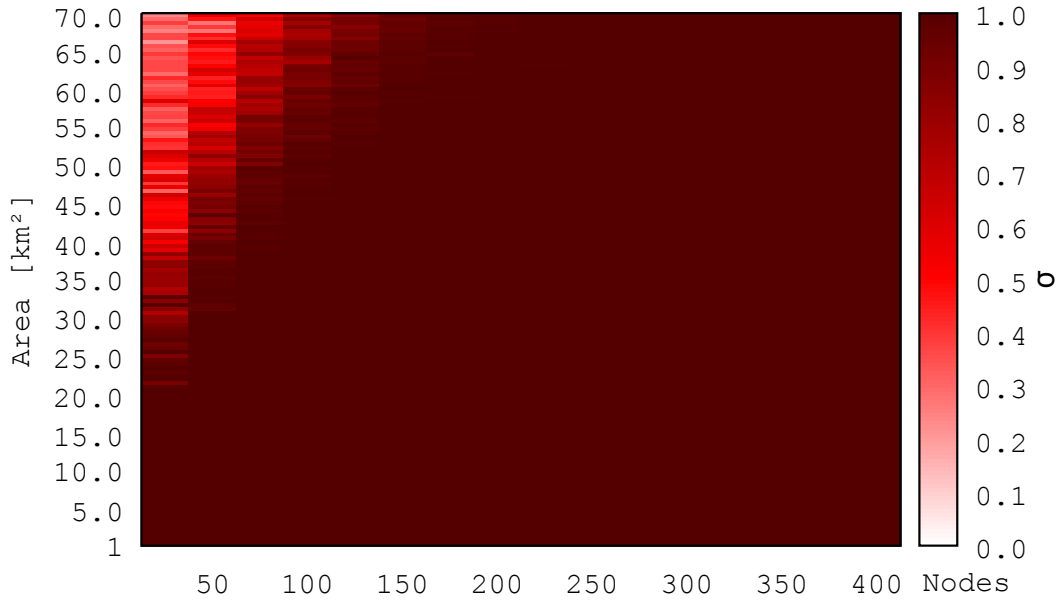
The results after 500 rounds of simulation with Bluetooth link settings for PR$_{0.4}$ and Random peer syncs are shown in Figures 4.7(a),(b) for ManhattanGrid mobility. For GaussMarkov mobility, this is shown in Figures 4.9(a),(b), respectively (cf. Section 4.1.2 for the creation of the synthetic mobility data). Each figure shows a heat



(a) ManhattanGrid: $\mathrm{PR}_{0.4}\ \Lambda_2^{0.1k}$.



(b) ManhattanGrid: RD $\Lambda_2^{0.1k}$.



(c) ManhattanGrid: unlimited $\Lambda_{\max}^{\infty}$.

Figure 4.7: *Synthetic mobility: final round global message spread σ with Bluetooth link settings in (a)–(b) for Prioritized ($PR_{0.4}\Lambda_2^{0.1k}$) and Random (RD $\Lambda_2^{0.1k}$) peer syncs, and unlimited link settings in (c) for $\Lambda_{\max}^{\infty}$ peer syncs.*

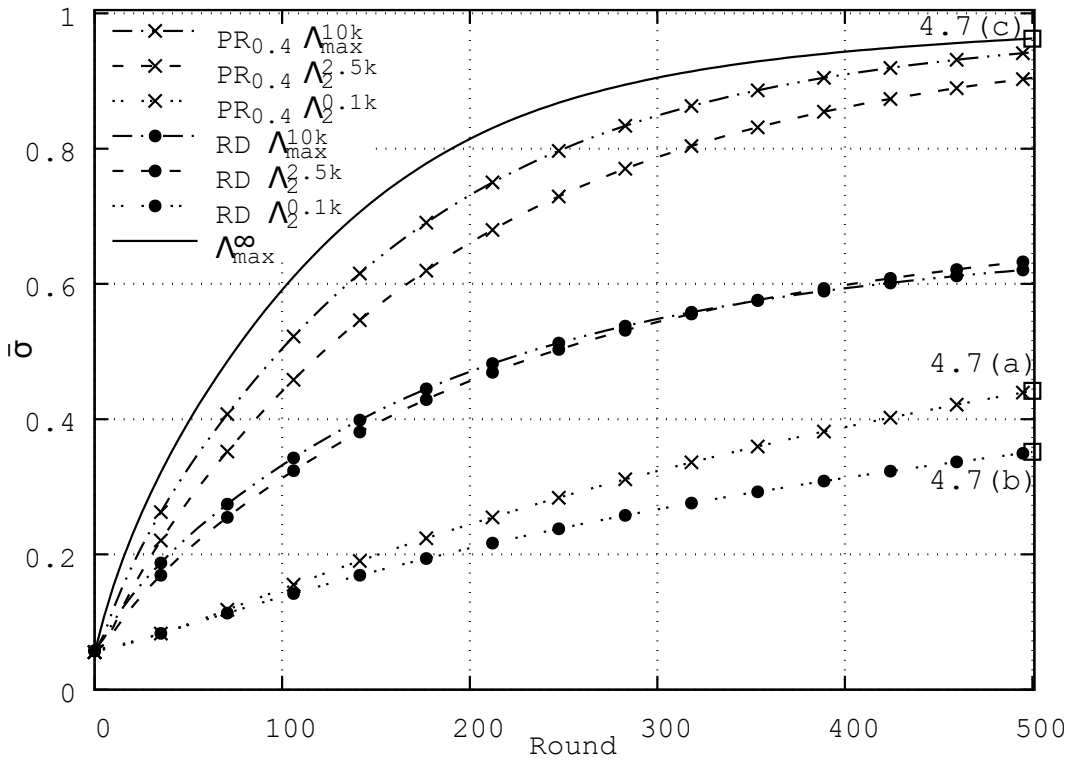
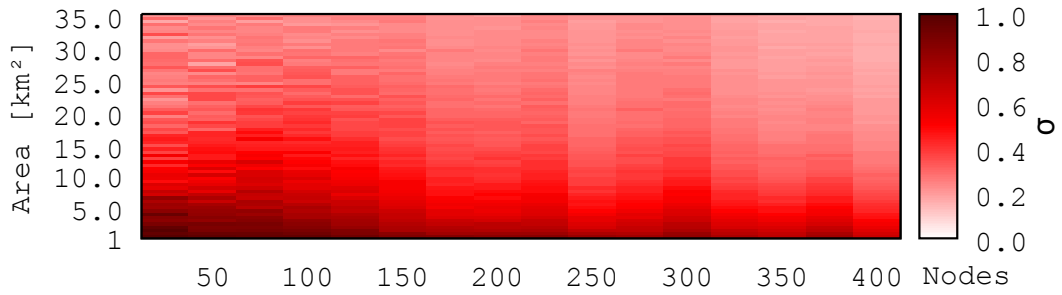


Figure 4.8: *ManhattanGrid synthetic mobility: mean global message spread $\bar{\sigma}$ with varied link settings for Prioritized (PR) and Random (RD) peer syncs and unlimited $\Lambda^{\infty}_{\max}$ link settings. The square-shaped points at round 500 named 4.7(a),(b),(c) show the $\bar{\sigma}$-values of the correspondingly captioned heat map Figures for ManhattanGrid.*

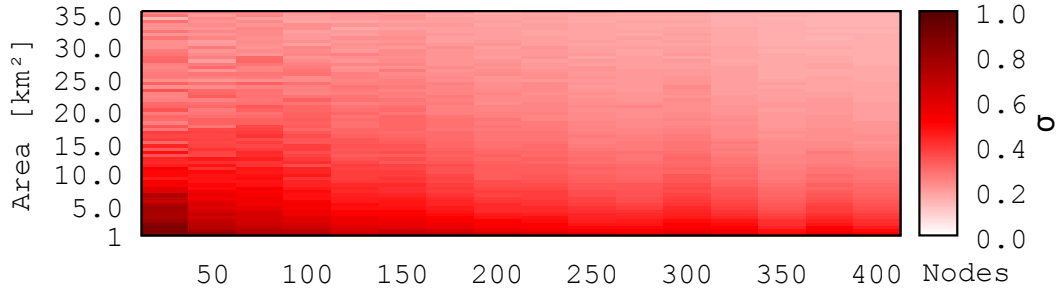
map of the final global message spread σ under a fixed sync strategy, for mobility inputs with increasing node number on the x-axis and increasing area on the y-axis.

Random syncs for ManhattanGrid achieves a message spread of 70% for 25 nodes up to an area of $25km^2$, and for 200 nodes for areas up to $5km^2$. GaussMarkov only achieves this message spread for 25 nodes up to $5km^2$, and for 100 nodes up to $3km^2$. $\mathrm{PR}_{0.4}$ shows higher message spreads: 70% spread is reached for 350 nodes up to $10km^2$ in ManhattanGrid, and in GaussMarkov for 125 nodes up to $5km^2$. GaussMarkov obtains a message spread close to 100% only for small areas and 25 nodes, whereas ManhattanGrid exhibits such a spread up to areas of $15km^2$ for 25 nodes, which then decreases to areas of up to $5km^2$ for 150 nodes. This shows again that the microblogging works in smaller areas under Bluetooth link settings for node numbers ranging from 25 to 150, but does not scale well for higher node numbers in these areas, since bandwidth is the limiting factor.

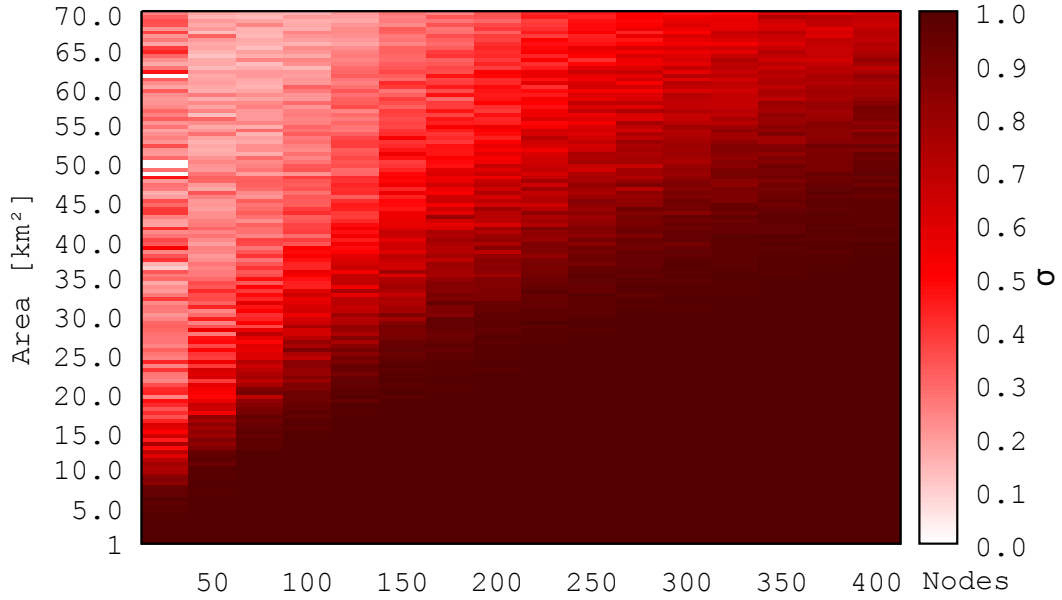
The more densely connected mobility of ManhattanGrid leads to higher message spread than the less structured mobility of GaussMarkov, and the selfish $\mathrm{PR}_{0.4}$ peer



(a) GaussMarkov: $PR_{0.4}\ \Lambda_2^{0.1k}$.



(b) GaussMarkov: RD $\Lambda_2^{0.1k}$.



(c) GaussMarkov: unlimited $\Lambda_{\max}^{\infty}$.

Figure 4.9: *Synthetic mobility: final round global message spread σ with Bluetooth link settings in (a)–(b) for Prioritized ($PR_{0.4}\ \Lambda_2^{0.1k}$) and Random (RD $\Lambda_2^{0.1k}$) peer syncs, and unlimited link settings in (c) for $\Lambda_{\max}^{\infty}$ peer syncs.*

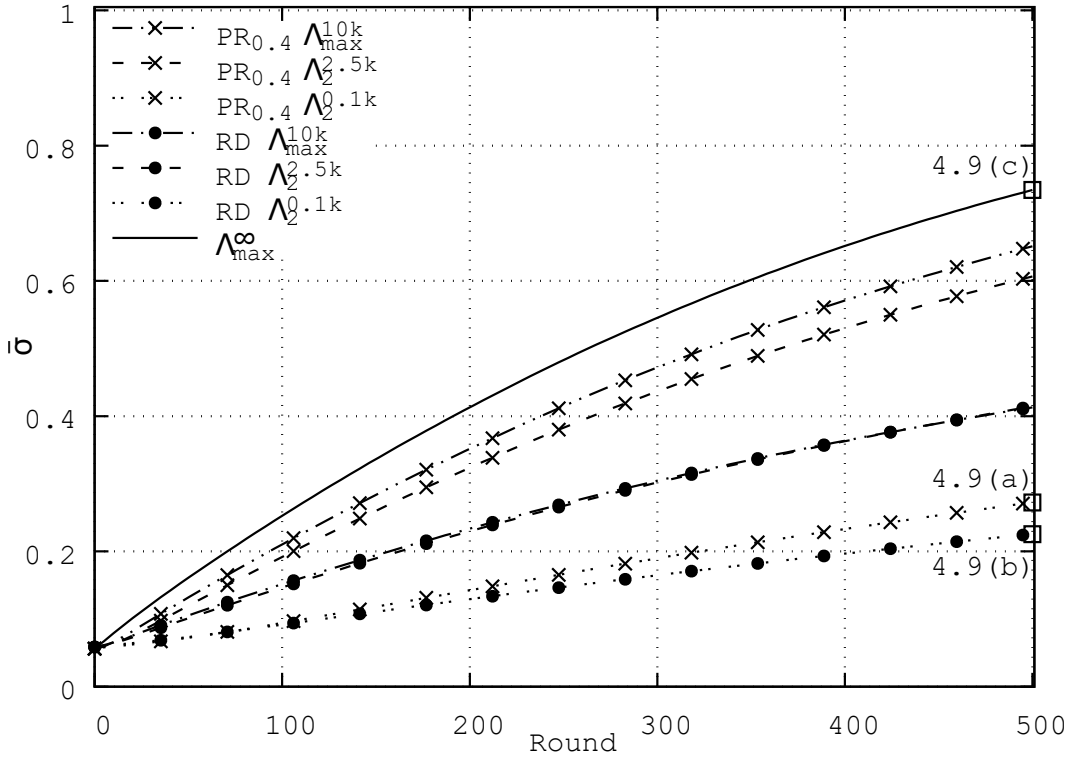


Figure 4.10: *GaussMarkov synthetic mobility: mean global message spread $\bar{\sigma}$ with varied link settings for Prioritized (PR) and Random (RD) peer syncs and unlimited $\Lambda^{\infty}_{\max}$ link settings. The square-shaped points at round 500 named 4.9(a),(b),(c) show the $\bar{\sigma}$-values of the correspondingly captioned heat map Figures for GaussMarkov.*

syncs similarly improve message spread over the Random peer syncs. Overall, the results are counter-intuitive: the observation of a degressive spread trend for increased node numbers within the same area and thus higher node connectivity that does not yield a better message spread is a sign for bandwidth as limiting factor beside the mobility of the nodes. To confirm this hypothesis, we perform experiments using unlimited bandwidth.

Under unlimited conditions the final results after 500 rounds of simulation are shown in Figure 4.7(c) for ManhattanGrid and in Figure 4.9(c) for GaussMarkov. The propagation of the messages is positively influenced for higher node numbers in the same area as anticipated, resulting in a progressive message spread trend. A spread close to 100% is reached in ManhattenGrid for 25 nodes up to areas of $20km^2$, constantly increasing, so that from 200 nodes onwards full spread is achieved in areas greater than $70km^2$. In GaussMarkov near 100% spread is reached in areas up to $7.5km^2$ for 25 nodes, and from 400 nodes onwards full spread is reached in areas greater than $50km^2$.

To approximate how well link settings other than the introduced Bluetooth setting can close up to the Unlimited setting, the hypothetical upper bound, a number of varied link settings have been run. Figures 4.8 and 4.10 show the arithmetic mean global message spread $\bar{\sigma}$ over time for the synthetic mobilities, defined as

$$\bar{\sigma} = \frac{\Sigma_{n \in N} \Sigma_{a \in A} \sigma_{n,a}}{|N|\,|A|},$$

with $N = \{25, 50, \ldots, 400\}$ being the set of node numbers, $A = \{1, 1.5, \ldots, 70\}$ the set of areas, and $\sigma_{n,a}$ the global message spread for the scenario with n nodes in an area of a km^2. For ManhattanGrid in Figure 4.8, the square-shaped points at round 500 named 4.7(a),(b),(c) correspond to the $\bar{\sigma}$-values of the heat map figures of the same name. For GaussMarkov in Figure 4.10, the square-shaped points analogously correspond to the $\bar{\sigma}$-values of the heat map Figures 4.9(a),(b),(c).

Overall, we see that, for $\mathrm{PR}_{0.4}$ peer syncs, the hypothetical upper bound can be reached by a margin of approximately 2% for ManhattanGrid with $\Lambda_{\max}^{10k}$ link settings (and a margin of approximately 6% with $\Lambda_2^{2.5k}$ link settings). For GaussMarkov, the margins amount to 8% and 12%, respectively. For Random (RD) peer syncs, the hypothetical upper bound cannot be reached this well for both mobilities, and an increase in link capacity from $\Lambda_2^{2.5k}$ to $\Lambda_{\max}^{10k}$ only yields a marginally better spread (and for ManhattanGrid the final values with $\Lambda_{\max}^{10k}$ link settings are slightly lower than with $\Lambda_2^{2.5k}$ settings). Random peer syncs result in nodes that indiscriminately re-send duplicates of the same messages. For a higher message spread nodes should therefore use Prioritized peer syncs.

In conclusion, we state that a major limiting factor for the propagation is the capacity of the transmission channel, and not only the empirical/synthetic mobility experienced by the nodes. The characteristics of the mobility patterns are an important factor, e.g., node degree and meetings per node per day. For real world scenarios the velocity of the nodes (and thus the resulting contact durations) has a significant impact on how the system works with a given technology: human movements show longer contact durations than vehicular movements, with the latter asking for technologies with fast connection establishment and high bandwidth. Especially partitions of the network are a real bottleneck for the conducted peer sync simulations, e.g., the Lausanne datasets with their strong bipartition. In general, it has been found that prioritizing messages yields a better message spread than random peer syncs and syncs with more than two peers per round does only yield better spread for datasets with node degrees higher than two, e.g., the Shanghai datasets.

4.3 Privacy and Security

The goals of confidentiality and anonymity are evaluated against a global passive adversary monitoring all network communication (peer syncs). For censorship-resistance,

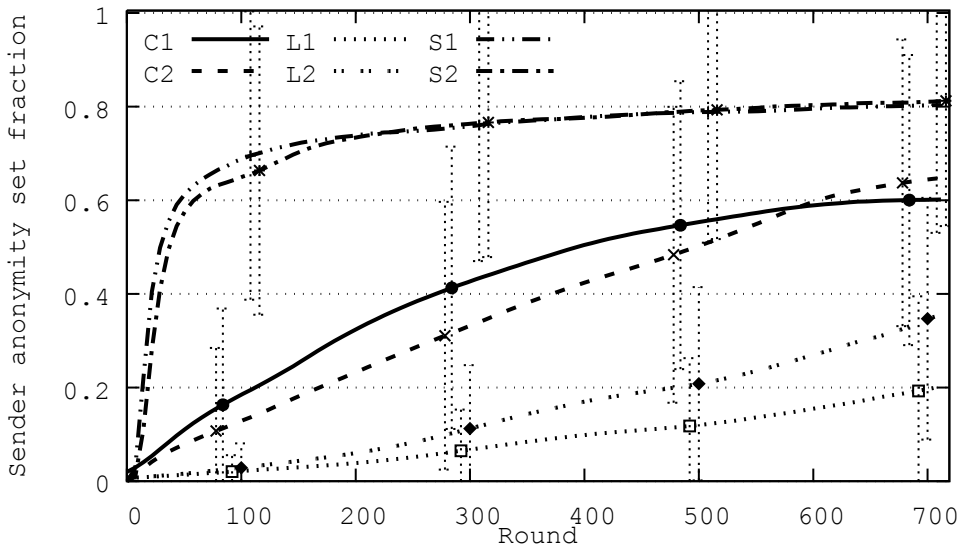


Figure 4.11: *Sender anonymity over time for empirical Congress (C1 and C2), Lausanne (L1 and L2), and Shanghai (S1 and S2) mobility: given that a received message has been created x rounds beforehand, what fraction of the total node number on average could have been the sender.*

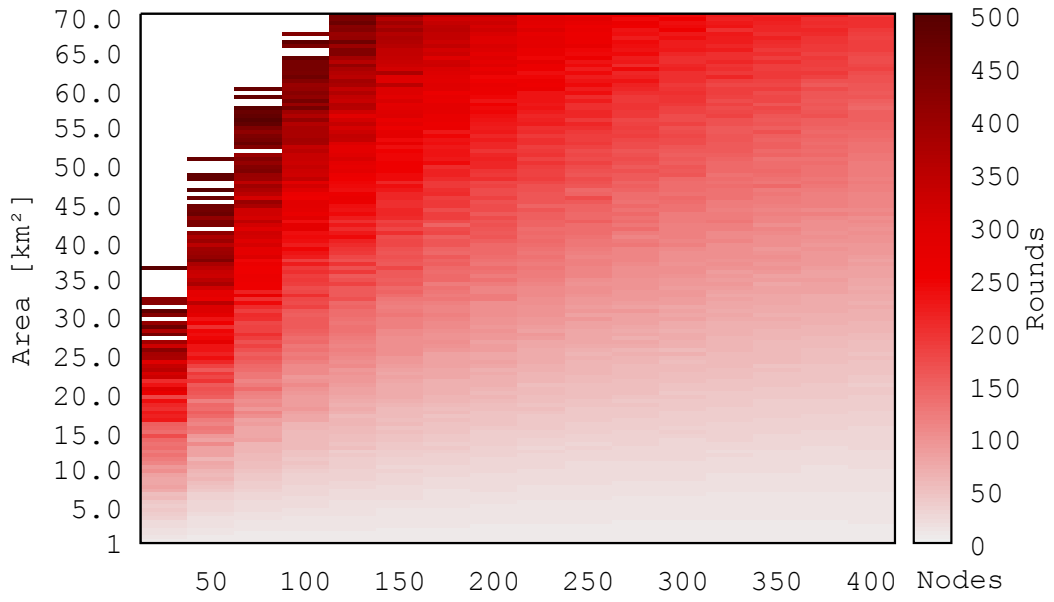
we assume an adversary that can jam or spam the system.

4.3.1 Privacy and Anonymity

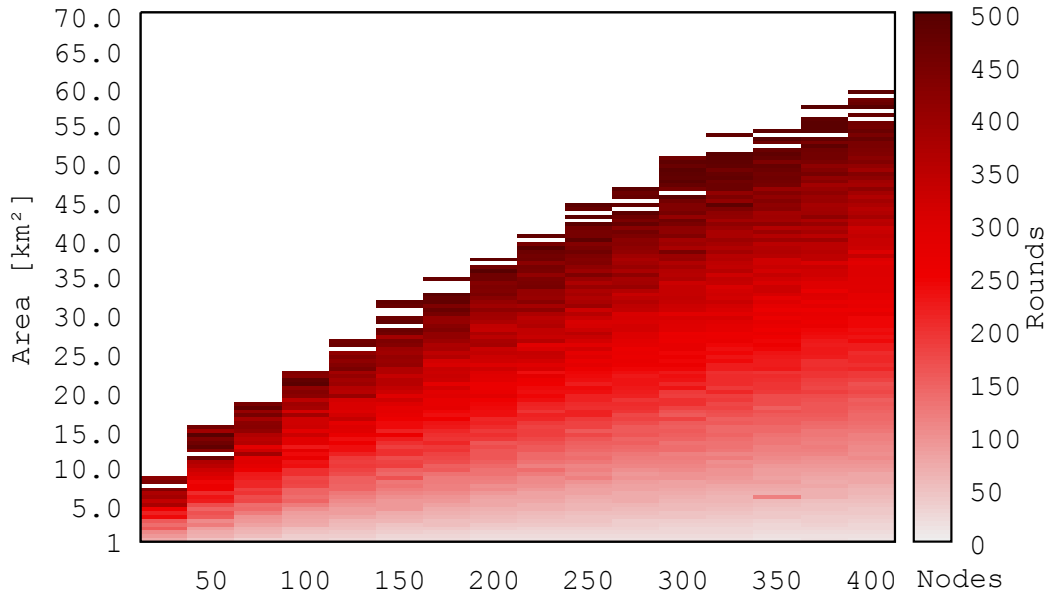
The P2P adversary introduced in Section 3.1.4 is not able to read any of the encrypted messages, thus the system keeps confidentiality. Receiver anonymity is achieved, since message content and group memberships also remain confidential from an adversary which is unable to compromise nodes, so that the P2P adversary can only hypothesize about possible senders and receivers of messages.

In order to experimentally evaluate sender anonymity, we assume the worst case, namely a P2P adversary with global visibility (instead of the assumed local visibility) who has monitored all peer syncs that took place, and who retrospectively tries to identify the set of possible senders of a message assuming that the message was created a specific number of rounds ago. The subsequently discussed figures are all based on a link setting with two peer syncs per round to illustrate the achievable levels of sender anonymity.

Figure 4.11 shows the sender anonymity over time for the empirical datasets. Sender anonymity set fraction signifies how many nodes on average could have sent a message to a receiving node, assuming knowledge of the adversary that this message has been created x rounds ago, e.g., for S2 60% of the 4445 nodes could have been the sender after 50 rounds, whereas for L2 only 20% of the 137 nodes could have been



(a) ManhattanGrid.



(b) GaussMarkov.

Figure 4.12: *Sender anonymity for synthetic mobility: the color indicates how many rounds ago a received message would have had to be created, so that 80% of the total node number on average could have been the sender – no color means this level of anonymity can not be reached in the given scenario within the observed time span.*

the sender after 700 rounds. Generally, we can observe that the underlying mobility highly determines how fast sender anonymity increases. The high connectivity of the Shanghai datasets thus yields a large possible anonymity set fraction after only a few rounds, and the Lausanne datasets – due to their bipartitioned nature – yield small anonymity set fractions, even after 700 rounds.

Figures 4.12(a) and 4.12(b) show a heat map for the synthetic datasets. The color indicates the number of rounds needed to achieve sender anonymity set fractions of 80% with respect to the total number of nodes; no color indicates levels of anonymity unobtainable within the illustrated upper bound of rounds, but possibly with a larger one. The more connected the nodes are, the faster a high level of anonymity can be reached. Overall, the achievable sender anonymity is closely linked to the underlying mobility.

4.3.2 Censorship-Resistance

Censorship-resistance means that an active adversary is unable to stop the propagation of messages based on their content, i.e., that in this case the system is not rendered dysfunctional in terms of achieved message spread. To evaluate censorship-resistance, we assume that the P2P adversary leverages his local visibility and internal participation (cf. Section 3.1.4) to either jam a fraction of the nodes per round or to inject spam messages.

Jamming is an attack used to stop the peer syncs between nodes. The adversary has the capability to limited local jamming, i.e., by the deliberate use of equipment that creates radio noise that disables all close-by peer syncs that take place. Jammed simulation runs are denoted by J_j, with $0 \leq j \leq 1$ being the fraction of all nodes which are each round randomly disabled from performing peer syncs.

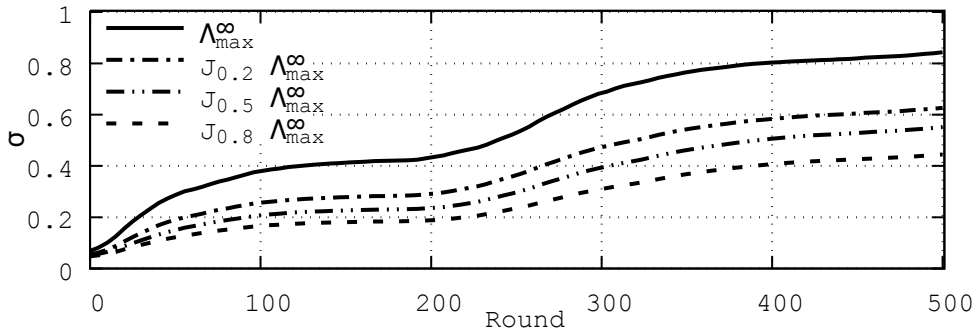
Selected message spread results with $J_{0.2}$, $J_{0.5}$ and $J_{0.8}$ simulation runs are shown in Figure 4.13(a) for the Congress2 empirical mobility and in Figure 4.13(b) for the ManhattanGrid synthetic mobility. The final round message spread of Fig. 4.13(a) of $J_{0.2}$ is about 22% lower than the unlimited ($\Lambda^{\infty}_{\max}$) one. The subsequent final round differences amount to approximately 8% and 10% for $J_{0.5}$ and $J_{0.8}$, respectively, totaling to a loss of around 40% in the $J_{0.8}$ with respect to the $\Lambda^{\infty}_{\max}$ run. The synthetic ManhattanGrid results shown in Fig. 4.13(b) exhibit a final round loss of approximately 2% from its unlimited run, the subsequent losses being around 5% and 15%, amounting to a 22% loss in $J_{0.8}$ with respect to $\Lambda^{\infty}_{\max}$. Jamming thus significantly lowers the spread and the Congress2 mobility is more susceptible to jamming. Yet, the microblogging system still exhibits sufficient message propagation.

Spamming is the injection of superfluous garbage messages into the network. The spamming nodes' intent is a Denial-of-Service attack on the microblogging system, since their bogus messages can not easily be filtered out and create network load,

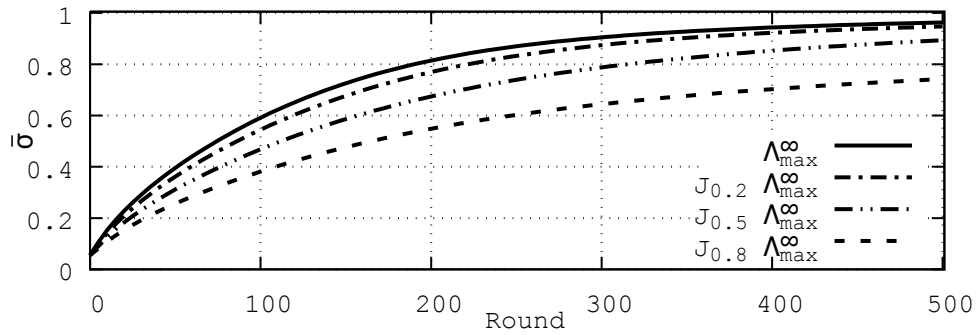
which in turn decreases spread of legitimate messages. Spammed simulation runs are denoted by S_s, with $0 \leq s \leq 1$ being the fraction of nodes that are spammers. We assume that all spamming nodes are not a member of any group and they peer-sync by filling their send buffers with new spam messages each round. These messages are indecipherable under any group key.

Selected message spread results are shown in Figure 4.14(a) for the Congress2 empirical mobility and in Figure 4.14(b) for the ManhattanGrid synthetic mobility. The Congress2 final round message spread in Fig. 4.14(a) is reduced from 74% to 47% when the regular $PR_{0.4}$ $\Lambda_2^{0.1k}$ is run with $S_{0.1}$ spamming (and reduced to 33% with $S_{0.4}$ spamming). The synthetic final round results in Fig. 4.14(b) only show losses of 8% and 5% with $S_{0.4}$ spamming in comparison to the regular $PR_{0.4}$ $\Lambda_2^{2.5k}$ and $PR_{0.4}$ $\Lambda_2^{0.1k}$ runs, respectively. If Random was used as peer sync strategy, the resulting message spreads would be lower.

In summary, the results of our conducted simulations show that the devised microblogging system can deal with a certain amount of jamming. However, the empirical Congress mobility dataset is much more susceptible to jamming than the

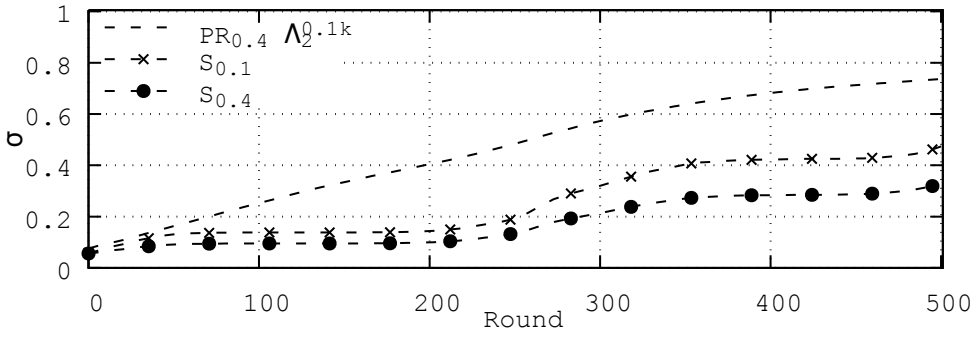


(a) Empirical Congress2 mobility.

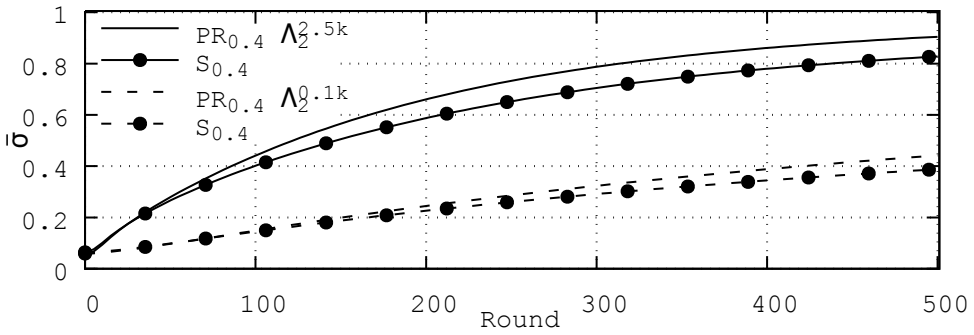


(b) Synthetic ManhattanGrid mobility.

Figure 4.13: *Jamming: selected jammed runs with unlimited (Λ^{∞}_{max}) peer syncs, with Fig. 9(a) showing global message spread results for Congress2 and Fig. 9(b) mean global message spread results for ManhattanGrid mobility.*



(a) Empirical Congress2 mobility.



(b) Synthetic ManhattanGrid mobility.

Figure 4.14: *Spamming: selected spam runs with Prioritized$_{0.4}$ ($PR_{0.4}$) peer syncs, with Fig. 10(a) showing global message spread results for Congress2 and Fig. 10(b) mean global message spread results for ManhattanGrid mobility.*

synthetic ManhattanGrid mobility dataset; here the former loses around 20% final message spread compared to the latter which loses only around 4% of the final message spread when a fifth of the communication is jammed. This disparity is even more prominent with respect to the robustness to spamming. In this case, the losses of the final message spread encountered are around 30% for the Congress dataset, even when only a tenth of the nodes spam with Prioritized$_{0.4}$ as sync strategy. Contrary to that, the synthetic ManhattanGrid mobility has more robustness to spamming, even if 40% of the nodes spam and Prioritized$_{0.4}$ is used as sync strategy. Overall, this shows that the system can not be fully stopped from propagating legitimate messages, but a stronger robustness to spamming would be desirable.

4.4 Discussion

Since node mobility and the technology used for peer syncs are the foundation of the microblogging system, these fundamental limitations of the system are addressed in

Section 4.4.1. Section 4.4.2 concludes the discussion and the findings of this chapter.

4.4.1 Limitations of Privacy-Enhanced Microblogging

In a decentralized system the message propagation is based on the mobility of the nodes in combination with the deployed peer-sync technology. If one group has several clusters of member nodes moving in disjoint network partitions, their messages will remain isolated within their respective partitions. The Lausanne dataset exhibits a case with a possible message spread of around 50% due to an underlying bipartitioned network structure. In such cases more node movement, determined by the underlying movement structures, or technologies that can conduct peer syncs over a larger distance are required to bridge partitions. However, one could also argue that the assignment to the groups in the Lausanne case was done without taking into account potentially existing social structures reflected in the movement, so that the resulting message spread was lower. Consequently, the system is considered to work better in small city boroughs, cf. the ManhattanGrid and Congress cases, and in scenarios with similar node movement patterns, i.e., the Shanghai cases.

4.4.2 Conclusion

It has been shown how a mobile distributed microblogging system allows to spread messages by direct, proximity-based peer syncs in real-world conditions. One identified bottleneck are the available technologies: with Bluetooth only sub-optimal message propagation is achievable, demanding for better suited technologies that have both higher bandwidth and faster connection establishment. Prioritized message propagation is preferable instead of Random syncs. With prioritization the microblogging system performs reasonably well under adversarial jamming and spamming, thus showing the desired censorship-resistance, whilst maintaining message confidentiality and anonymity.

Chapter 5

Private Set Intersection Peer Syncs

This chapter investigates whether instead of prioritized peer syncs, where the nodes draw a certain percentage of the messages to be put in the send buffer from groups they are a member of, it is beneficial for message spread if the nodes use Private Set Intersection. By the use of Private Set Intersection the two nodes in a peer sync first privately determine their common groups, and then prioritize their common group messages during peer syncs, i.e., put in their send buffer first group messages of groups both nodes are a member of. The hypothesis is, that these more target-oriented prioritized peer syncs with Private Set Intersection lead to a higher message spread of the system globally.

This chapter starts with a presentation of the cryptographic primitive of Private Set Intersection in Section 5.1. In particular, the on-device implementation of a specific Diffie-Hellman-based protocol is motivated based on the scenario requirements, including performance benchmarks of actual protocol runs. To evaluate the hypothesis that deploying Private Set Intersection during peer syncs is beneficial for message spread, we experimentally investigate it with simulations. We present the simulation results with Private Set Intersection peer syncs in Section 5.2, before we draw a conclusion in Section 5.3.

5.1 Private Set Intersection

Private Set Intersection (PSI) is a cryptographic protocol that lets two parties who both have their own sets of elements determine the elements they both have in common (the intersection of their two sets), without revealing any elements to the other party that are not in the intersection. Consider the case of Alice with her set of elements $A = \{1, 2, 3\}$ and Bob with his set of elements $B = \{2, 3, 4\}$. On Alice's input A and Bob's input B, the protocol computes $A \cap B$ and outputs the set $\{2, 3\}$ to both parties. By running the protocol, Alice does not get to know anything about the element 4 in Bob's set of elements, while Bob does not get to know the element 1

in Alice's set of elements. With standard PSI protocols, the only knowledge gained besides the intersection of elements is the number of elements each party has in its set, i.e., $|A| = 3$ and $|B| = 3$. There are more complex PSI protocols that hide the cardinality, meaning that no information leakage is revealed on the input set sizes of Alice and Bob. However, a simple way to achieve this property is to set an upper bound of elements per set and to let both parties pad their sets with dummy elements until the upper bound of elements is reached. Then both parties have input set sizes of the same size – the upper bound – and no information is leaked on their actual input set sizes.

5.1.1 Protocols for Private Set Intersection

The protocols of PSI can be grouped into three classes: Garbled Circuit based protocols, Oblivious Transfer based protocols, and Public Key cryptography based protocols. Some protocols require a Trusted Third Party (TTP), and one could argue that they constitute another class of PSI protocols. However, the requirements of the microblogging solution described in this thesis do not allow for a TTP in the case of PSI peer-syncing, and therefore protocols with a TTP are omitted in this study. The remainder of this section shortly outlines these three classes of protocols based on informal definitions and briefly compares them.

Garbled Circuit Based PSI Protocols

Garbled circuit or generic secure computation protocols are based on either Boolean or arithmetic circuits. [127] These circuits enable the secure computation of arbitrary functions. They are mostly based on either Yao's garbled circuit protocol [134] or the Goldreich–Micali–Wigderson protocol [53] in the case of Boolean circuits, and on the Ben-Or–Goldwasser–Wigderson [9] or the Chaum–Crépeau–Damgård [23] protocols in case of arithmetic circuits. Numerous PSI protocols based on garbled circuits have been proposed, and they experienced significant performance improvements over time. [98] Recently, in [62] several Boolean circuits have been proposed: the simple bitwise-AND protocol uses a bit-vector representation of the input sets and is instantiated by a binary AND gate 2^s times, with s being the size of the set elements in bits. Even though the complexity of this protocol grows exponentially with s, the performance has been found to be best for values of $s \leq 16$. The pairwise-compare protocol uses a circuit that computes pairwise comparisons of the elements in the two input sets. It has a worst-case complexity of $\mathcal{O}(n^2)$, with n being the sizes of the input sets, thus rendering it impractical for larger input sets. The sort-compare-shuffle protocol is based on the idea that both parties first privately and locally sort their sets into a single sorted list, then comparing all neighboring elements for equality, and finally shuffling the intersecting elements to hide any leakage that could

be obtained from the resulting positional information. This protocol has a complexity of $\mathcal{O}(n \log n)$, which is better than that of the pairwise-compare protocol and is thus better suited for larger input sets. In [99] Pinkas et al. present a new circuit-based PSI protocol which leverages new hashing algorithms and which only requires the computation of $\omega(n)$ comparisons.

Oblivous Transfer Based PSI Protocols

This class of protocols makes use of the cryptographic primitive of Oblivious Transfers (OT). The OT-primitive is, generally speaking, a two-party protocol, in which Alice has a set of n messages $a_1,\ a_2,\ \cdots,\ a_n$ with indexes $1 \leq i \leq n$ and Bob queries an index i from Alice at his own discretion. At the end of the protocol interaction between the two parties Bob receives a_i without gaining any information about Alice's other messages, whilst Alice learns nothing about Bob's queried index i (cf. [135]).

A way to efficiently create a large number of OTs is referred to as an OT extension and was initially published in [64]. The basic idea is to instead of computing a large number of OTs with slow public key operations, it is feasible to first compute a smaller number of base OTs. From these base OTs a larger number of OTs can be derived solely by faster symmetric key operations, making OT extremely inexpensive in practice. Recently, OT-based PSI protocols that make use of Bloom filters [12] over an OT extension [72] have been proposed in [40, 73, 98].

Public Key Cryptography Based PSI Protocols

Asymmetric or public key cryptography is characterized by each user to have an unique key pair, i.e., a public key that is usually not a secret and published, and a secret or private key only known to the user, and that is generally kept as a secret by the user owning the key pair. Public key based PSI protocols make use of public key cryptography based protocols such as ElGamal [48], Pallier [92] and RSA [107], or the Diffie-Hellman (DH) key exchange protocol [38]. PSI protocols based on ElGamal are for example found in [17, 32, 59, 60], protocols based on the Paillier cryptosystem can be found in [47, 71], protocols based on the RSA cryptosystem in [29, 31], and protocols based on the DH key exchange in [63, 85].

Comparison of PSI Protocol Classes

Garbled circuit based PSI protocols can be built relatively easily by a programmer with knowledge of circuits. Essentially, this is achieved by the use of software packages specifically made for the construction of garbled-circuit protocols [81, 82]. The task at hand is then reduced to the design of a circuit that evaluates the function to be computed. Garbled circuit protocols are also more modular than the other

two classes of protocols, and the functionality of the protocols is simply extendable, without the need to change the protocol and without affecting its security. In the case of PSI, it is straightforward to adapt a given PSI protocol to support additional computations via the extension of the circuit with the wanted computational functionality, e.g., the size of the intersection or the sum of the values of the items that reside in the intersection.

For the other two classes –oblivious transfer and public key based– the task of building a PSI protocol is more difficult and regularly requires the knowledge of a cryptographer, who has to design and implement the PSI protocol from scratch. Also, extending these special purpose protocols for additional computations is more difficult and may possibly come at the cost of losing efficiency.

Since PSI is an active research field, many proposals for PSI protocols exist. Generalized statements on the efficiency and performance of the vast number of suggested protocols are thus difficult to make, which is on the one hand further complicated by the lack of implementations of many protocol designs. On the other hand, the protocols that have been implemented and evaluated, were often benchmarked under varying assumptions and observations, e.g., the protocol has been tested for the total runtime while ignoring other important aspects such as the communication complexity [100].

In [101] a cross-evaluation of selected PSI protocols belonging to the three classes summarized above has been conducted. The communication and computation complexity are among the main criteria used to evaluate the performance of PSI protocols and all protocols have been implemented using the same programming language. Also, the same cryptographic libraries have been used and the hardware remained the same, thus allowing a fair comparison of the tested protocols.

Some of the main findings are that the communication complexity of the garbled circuit based protocols is by far the highest among all tested protocols. The garbled circuit protocols also have high computation complexity, even though their runtime is smaller than the public key based protocols. The public key based protocols require costly public key operations and they do not scale well with increased input set sizes; however they have the best communication complexity of all the tested protocols. Finally, the tested oblivious transfer based protocols have the smallest computation complexity and are second best with respect to communication complexity, i.e., they are up to ten times faster and have a communication complexity which is only about two times higher than the tested public key based protocols (cf. [101]).

5.1.2 On-device Implementation of DH-PSI Protocol

The requirements of a PSI protocol useful in our setting are, that it is a two-party protocol between parties A and B without any participation of a Trusted Third

Party (TTP), not even during a setup phase before the actual protocol execution. Also, the adversary model is the honest but curious model, meaning that the participants are curious to find out all they can about the secrets of each other, but they do not deviate from the specified protocol run in any way. The input set sizes are relatively small, i.e., with n being the cardinality of the input set sizes, its distribution would look approximately as follows, based on the observations made in [74]:

$$\text{size of } n \text{ is } \begin{cases} 1 \leq n \leq 100, & \text{in } 99\% \text{ of the cases,} \\ 100 < n \leq 1,000, & \text{in } 0.99\% \text{ of the cases,} \\ 1,000 < n \leq 10,000, & \text{in } 0.01\% \text{ of the cases.} \end{cases}$$

This means that in 99 out of 100 cases, the input set size is smaller than 100, and only in 1 out of 10,000 cases is the input set size larger than 1,000 but smaller than 10,000. The protocol does not need to have the private set size hiding property, a property that can also be achieved by dummy padding, but the protocol execution should be fast on current smartphones.

Implementation of a DH-based PSI Protocol

We chose the protocol described in [63, 85] for inclusion into the private microblogging solution, also in agreement with the recommendations made in [100]. The decision to use this protocol is because of its fast computation runtime for the given set sizes n and the best communication complexity of all tested PSI protocols in [100]. Both complexities scale linearly with the set sizes, i.e., $\mathcal{O}(n)$. The computation complexity is costly due to the public key operations, but since the input set sizes are comparably small, the runtime is exptected to be acceptably fast. More important is the protocol's low communication complexity that is needed in the delay-tolerant networking setting in which nodes may only be in communication range for a limited period of time.

The protocol itself is DH-based and its basic matchmaking form, i.e., both parties only have one secret element and they check if they are equal, has been published in [85]. It involves only one secret by each party and it is subsequently shown as a protocol between the two parties A and B. All calculations are performed in the multiplicative group $\mathbb{Z}_p$. Party A possesses a as a secret, and party B possesses b as a secret; both a and b are generators of the multiplicative group $\mathbb{Z}_p$. Both parties are required to choose secretely and uniformly at random another element: A chooses $r \in \mathbb{Z}_p$ and B chooses $s \in \mathbb{Z}_p$.

1. $A \rightarrow B : a^r$
2. $B \rightarrow A : b^s$
3. $A \rightarrow B : (b^s)^r (= b^{rs})$
4. $B \rightarrow A : (a^r)^s (= a^{rs})$

Similar to the Diffie-Hellman protocol for secure key exchange, it relies on the observation that $(X^Y)^Z = X^{YZ} = X^{ZY} = (X^Z)^Y$. If A and B have a matching element, both parties can verify this by the following calculations: A takes the value received in step 4 and checks if it equals to the value A has send to B in step 3, and B has to analogously check if the value received in step 3 equals to the value B has sent in step 4. If this is the case, they have a matching value.

On-device Performance of the DH-based Protocol

The simple idea presented above can be extended into a practical PSI protocol, so that it can accommodate the comparision of an arbitrary number of elements instead of only one element. Instead of using the plain secrets as input, the hashed values of the secrets are used as input. This is useful to map potentially large secrets to a smaller value. Party A in this case has a_{nA} secrets $a_1, a_2, \ldots, a_{nA}$, and party B has b_{nB} secrets $b_1, b_2, \ldots, b_{nB}$, all of which are used in their hashed form only. The protocol is the following:

1. $A \rightarrow B : H(a_1)^r, H(a_2)^r, \ldots, H(a_{nA})^r$
2. $B \rightarrow A : H(b_1)^s, H(b_1)^s, \ldots, H(b_{nB})^s$
3. $A \rightarrow B : H(b_1)^{rs}, H(b_2)^{rs}, \ldots, H(b_{nB})^{rs}$
4. $B \rightarrow A : H(a_1)^{rs}, H(a_2)^{rs}, \ldots, H(a_{nA})^{rs}$

To check for the matching items, party A has to check which of the values received from B in step 4 are equal to the values it has sent to B in step 3; conversely, party B has to check for the values received from A in step 3 which equal to the values it has sent to A in step 4.

This protocol has been tested on a *Samsung Galaxy S5 mini* with respect to computation time, the results of which are shown in Table 5.1. The most costly operations are clearly the modular exponentiations that need to be computed in steps 1 and 3 of the protocol. They make up for almost all the computation time, whereas the computation time needed for the initial hashing of the input elements and the matching of the equal items at the end of the protocol run constitute only for a small fraction of the total runtime. Note that the runtime of steps 2 and 4 is

Number of input elements	10	100	1000
Hashing (SHA-256) runtime in ms	0,571	5,875	134,245
Step 1. runtime in ms	25,571	242,574	2 737,058
Step 3. runtime in ms	25,253	260,048	2 419,055
Intersection (matching) runtime in ms	0,214	0,727	2,417
Total runtime in ms	51,609	509,224	5 292,775

Table 5.1: *Evaluation of PSI-protocol: average computation runtimes for the different steps needed during the protocol run and average total runtime in milliseconds.*

essentially the same as the one for steps 1 and 3 of the protocol and therefore not listed in the table.

Table 5.1 shows that the protocol is practical to use in case of small to medium sets with total runtimes of about a twentieth of a second for input set sizes of 10 elements and about half a second for input set sizes of 100 elements. For large input set sizes of 1000 elements the runtime amounts to approximately 5.3 seconds which is still practical.

5.2 PSI-based Peer Sync Simulation Runs

The goal of this section is to evaluate whether the use of a peer-sync strategy based on PSI improves message distribution. During a peer sync, two nodes first perform PSI to determine common groups, i.e., groups where both are members. They then fill a portion of their send buffer with common group messages of both nodes. This new form of peer syncs is denoted as PSI_p, with p being the fraction of send slots which are reserved for common group messages. A node first uniformly at random selects from the common group messages and if this is not able to fill all reserved slots it uniformly at random selects from all other group messages. As for prioritized syncing, the remaining fraction $1-p$ is filled randomly with messages of other groups.

5.2.1 Performance of PSI Syncs on Empirical Data

The global message spread results for $\text{PSI}_{0.4}$ peer syncs for the empirical mobility datasets are subsequently discussed in comparison to the $\text{Prioritized}_{0.4}$ and Random peer syncs, where the link settings are adjusted to Bluetooth, i.e., $\Lambda_2^{0.1k}$. To give an idea of the best possible spread, the upper bound for unlimited link settings, i.e., $\Lambda_{\max}^{\infty}$, is included as well (as in Section 4.2.2).

For the Congress1 and Congress2 empirical mobility datasets the global message spreads over time are shown in Figure 5.1(a) and Figure 5.1(b), respectively. For

the Congress1 empirical mobility dataset in Figure 5.1(a) the global message spread with $PSI_{0.4}$ syncs is evolving similary to that with $PR_{0.4}$ syncs and is in the final round only approximately 6% lower than the $PR_{0.4}$. The situation looks different for the Congress2 empirical mobility dataset in Figure 5.1(b). Here, the $PSI_{0.4}$ curve is evolving similar to the unlimitied $\Lambda^{\infty}_{\max}$ curve, albeit with a much lower global message spread. Also, the message spread is lower than that of the $PR_{0.4}$ curve, having a message spread in the final round that is approximately 18% lower than that of $PR_{0.4}$, but still about 24% higher than that of RD syncs.

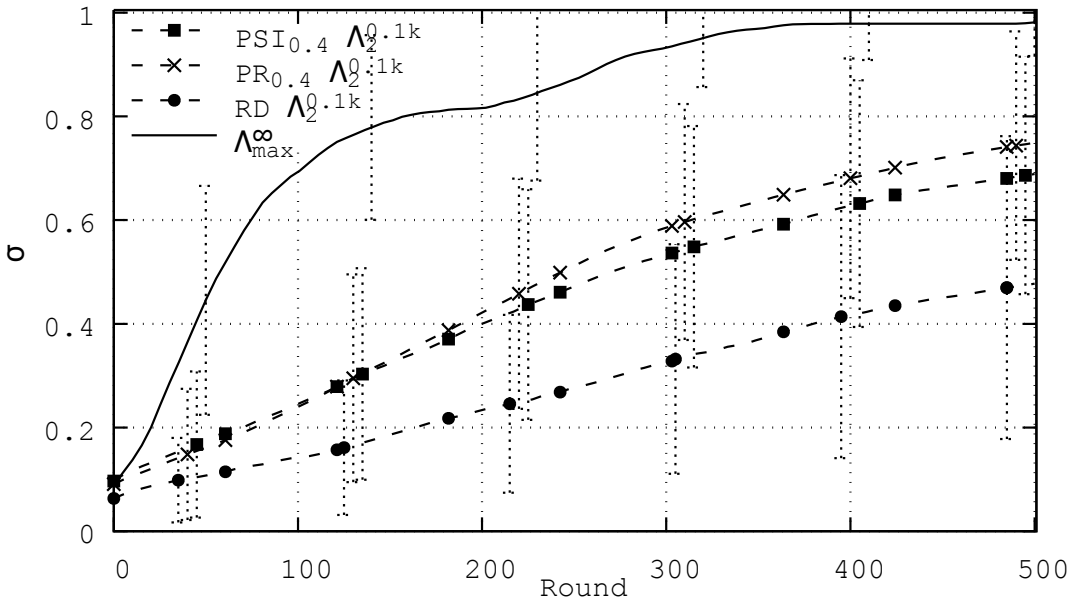
For the Lausanne1 and Lausanne2 empirical mobility datasets the global message spreads over time are shown in Figure 5.2(a) and Figure 5.2(b), respectively. The message spread in the datasets reached with Bluetooth link settings $\Lambda_2^{0.1k}$ are generally not very high. In Lausanne1 the spread with $PSI_{0.4}$ syncs is slightly lower than that of the $PR_{0.4}$ and the RD syncs. However, the opposite is the case in the Lausanne2 mobility dataset: here the global message spread with $PSI_{0.4}$ syncs is slightly higher than that with $PR_{0.4}$ and RD syncs. Note that in all cases the simulation results were averaged over five runs, thus preventing outliers.

For the Shanghai1 and Shanghai2 empirical mobility datasets the global message spreads over time are shown in Figure 5.3(a) and Figure 5.3(b), respectively. In the Shanghai1 dataset the global message spread with $PSI_{0.4}$ syncs is similar to the message spread with $PR_{0.4}$ syncs and slightly above the message spread with RD syncs. For Shanghai2 the results with $PSI_{0.4}$ syncs are slightly below the message spread with $PR_{0.4}$ syncs, and they have about the same message spread as with RD syncs.

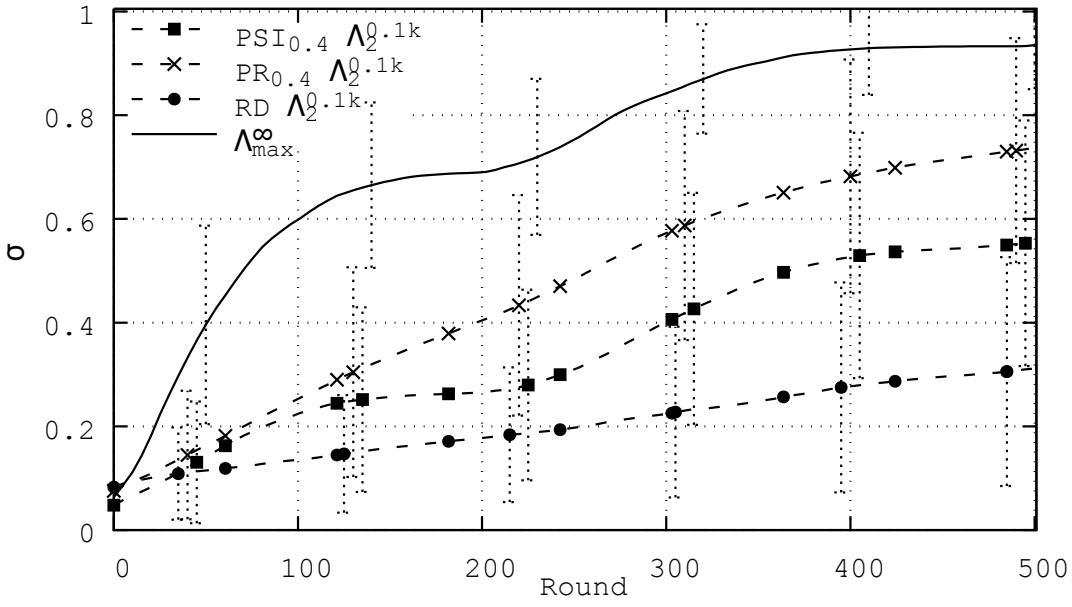
5.2.2 Performance of PSI Syncs on Synthetic Data

In order to have a broader set of results and to test boundaries, we also run simulations with PSI syncs on synthetic mobility data. The mean global message spreads $\bar{\sigma}$ for synthetic $PSI_{0.4}$ peer syncs over time are shown in Figure 5.4 for the ManhattanGrid synthetic mobility datasets and in Figure 5.5 for the GaussMarkov synthetic mobility datasets.

For the ManhattanGrid mobility in Figure 5.4, the results for $PSI_{0.4}$ $\Lambda_2^{0.1k}$ peer syncs in round 500 are approximately 2% higher than the PR $\Lambda_2^{0.1k}$ peer-sync results, and approximately 11% higher than the RD $\Lambda_2^{0.1k}$ peer-sync results. For the higher link capacity setting of the $PSI_{0.4}$ $\Lambda_2^{2.5k}$ peer syncs a mean global message spread $\bar{\sigma}$ of 77% is achieved in round 500, and with $PSI_{0.4}$ $\Lambda^{10k}_{\max}$ peer syncs the value of $\bar{\sigma}$ in round 500 amounts to 73%, meaning that the higher link capacity leads to a lower message spread of approximately 4%. Furthermore, the $PSI_{0.4}$ round 500 results for $\Lambda_2^{2.5k}$ and $\Lambda^{10k}_{\max}$ peer syncs are significantly lower than their corresponding $PR_{0.4}$ peer-sync results (13% and 21% lower for $\Lambda_2^{2.5k}$ and $\Lambda^{10k}_{\max}$, respectively), but their

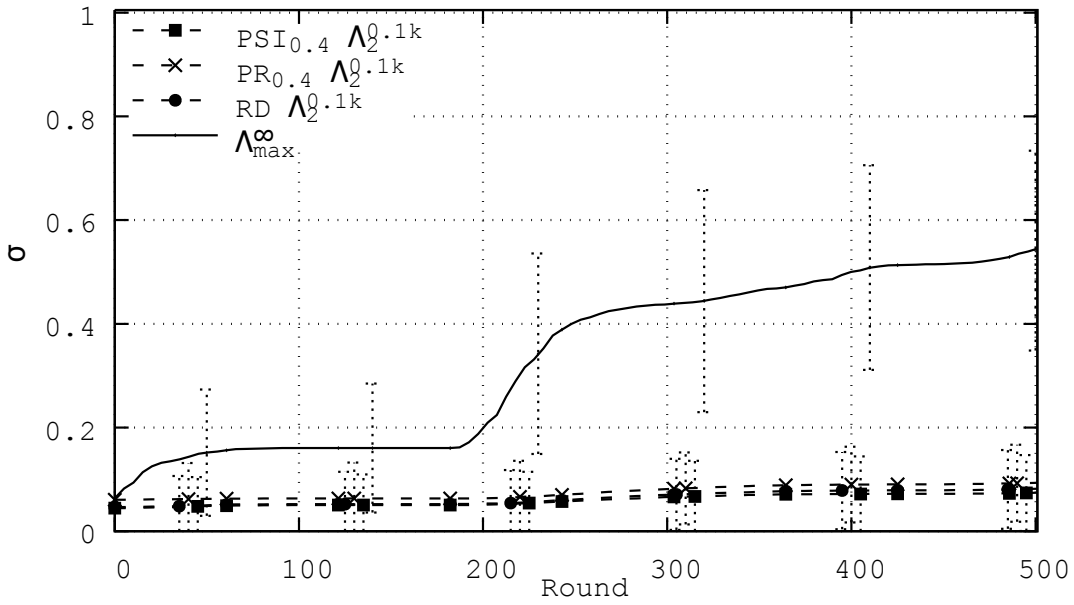


(a) Congress1.

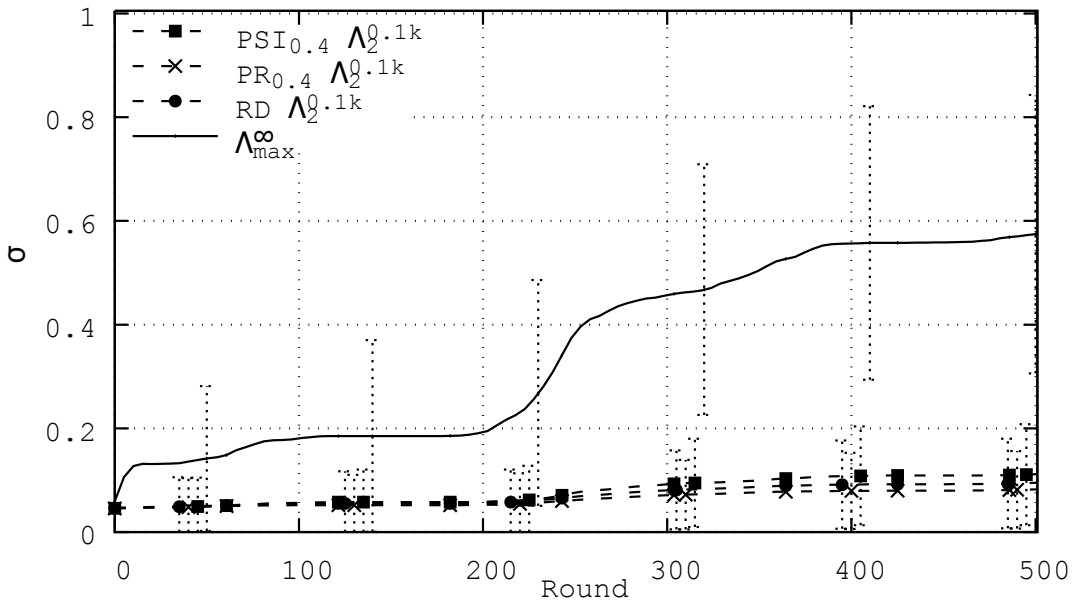


(b) Congress2.

Figure 5.1: *Congress empirical mobility: global message spread σ averaged over five runs with Bluetooth link settings for PSI ($PSI_{0.4}\Lambda_2^{0.1k}$), Prioritized ($PR_{0.4}\Lambda_2^{0.1k}$) and Random (RD $\Lambda_2^{0.1k}$) peer syncs, and unlimited link settings for $\Lambda_{\max}^{\infty}$ peer syncs.*

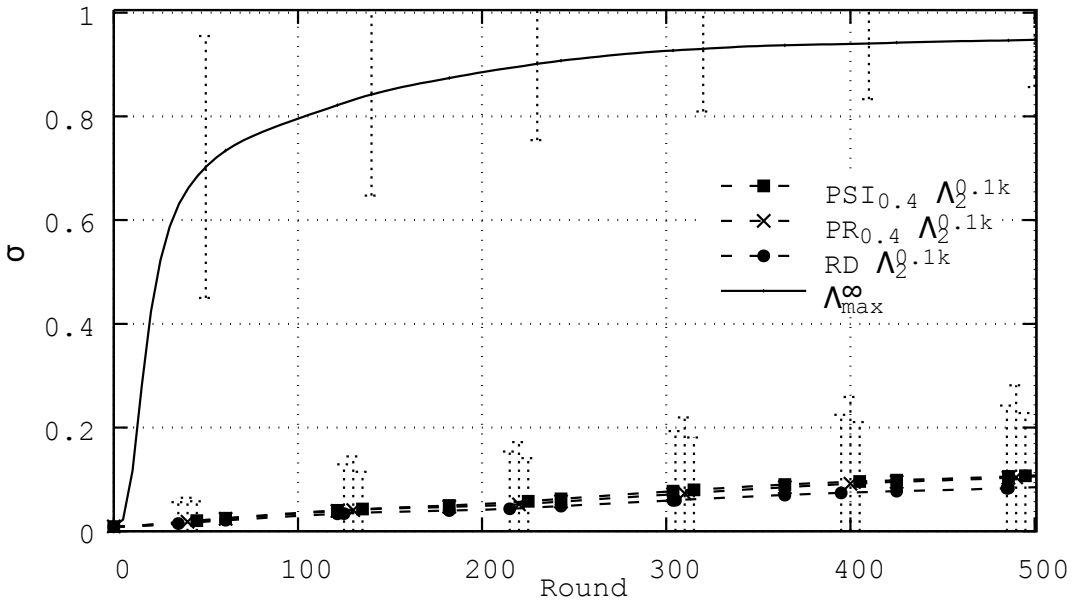


(a) Lausanne1.

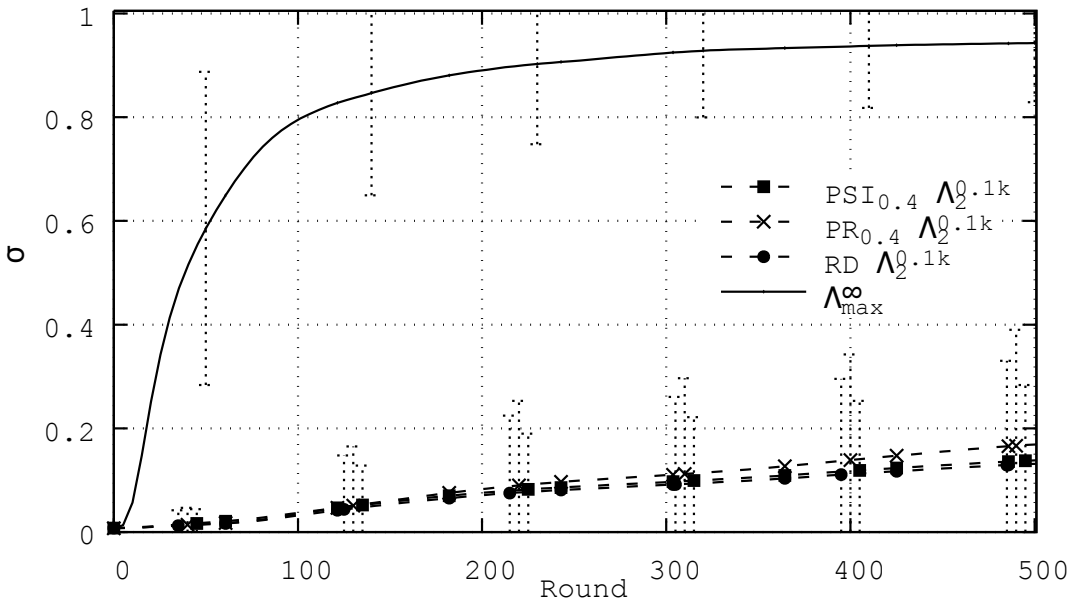


(b) Lausanne2.

Figure 5.2: *Lausanne empirical mobility: global message spread σ averaged over five runs with Bluetooth link settings for PSI ($PSI_{0.4}\Lambda_2^{0.1k}$), Prioritized ($PR_{0.4}\Lambda_2^{0.1k}$) and Random (RD $\Lambda_2^{0.1k}$) peer syncs, and unlimited link settings for $\Lambda_{\max}^{\infty}$ peer syncs.*



(a) Shanghai1.



(b) Shanghai2.

Figure 5.3: *Shanghai empirical mobility: global message spread σ averaged over five runs with Bluetooth link settings for PSI ($PSI_{0.4}\Lambda_2^{0.1k}$), Prioritized ($PR_{0.4}\Lambda_2^{0.1k}$) and Random (RD $\Lambda_2^{0.1k}$) peer syncs, and unlimited link settings for $\Lambda_{\max}^{\infty}$ peer syncs.*

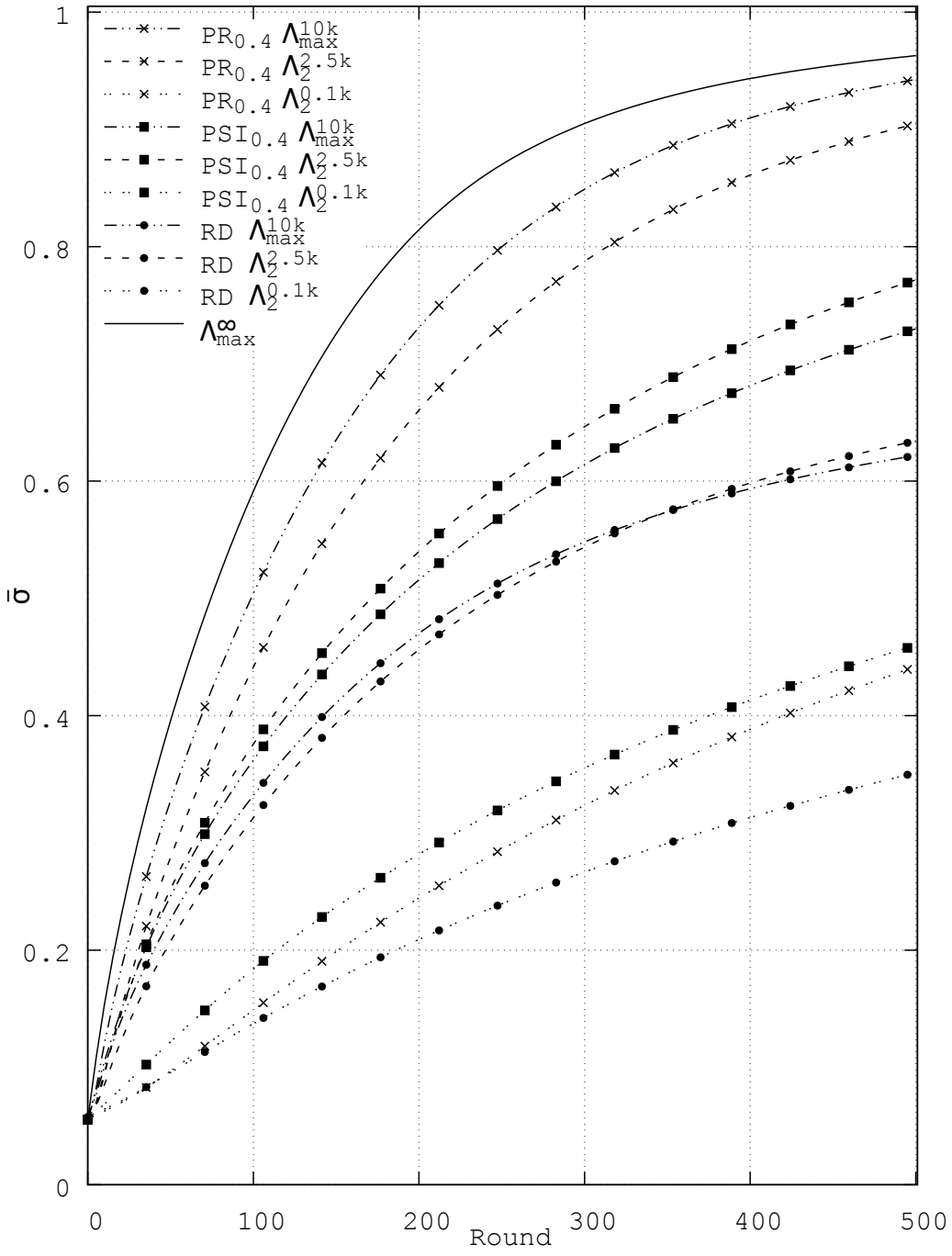


Figure 5.4: *ManhattanGrid synthetic mobility: mean global message spread $\bar{\sigma}$ with varied link settings for PSI, Prioritized (PR) and Random (RD) peer syncs, and unlimited link settings for $\Lambda^{\infty}_{\max}$ peer syncs.*

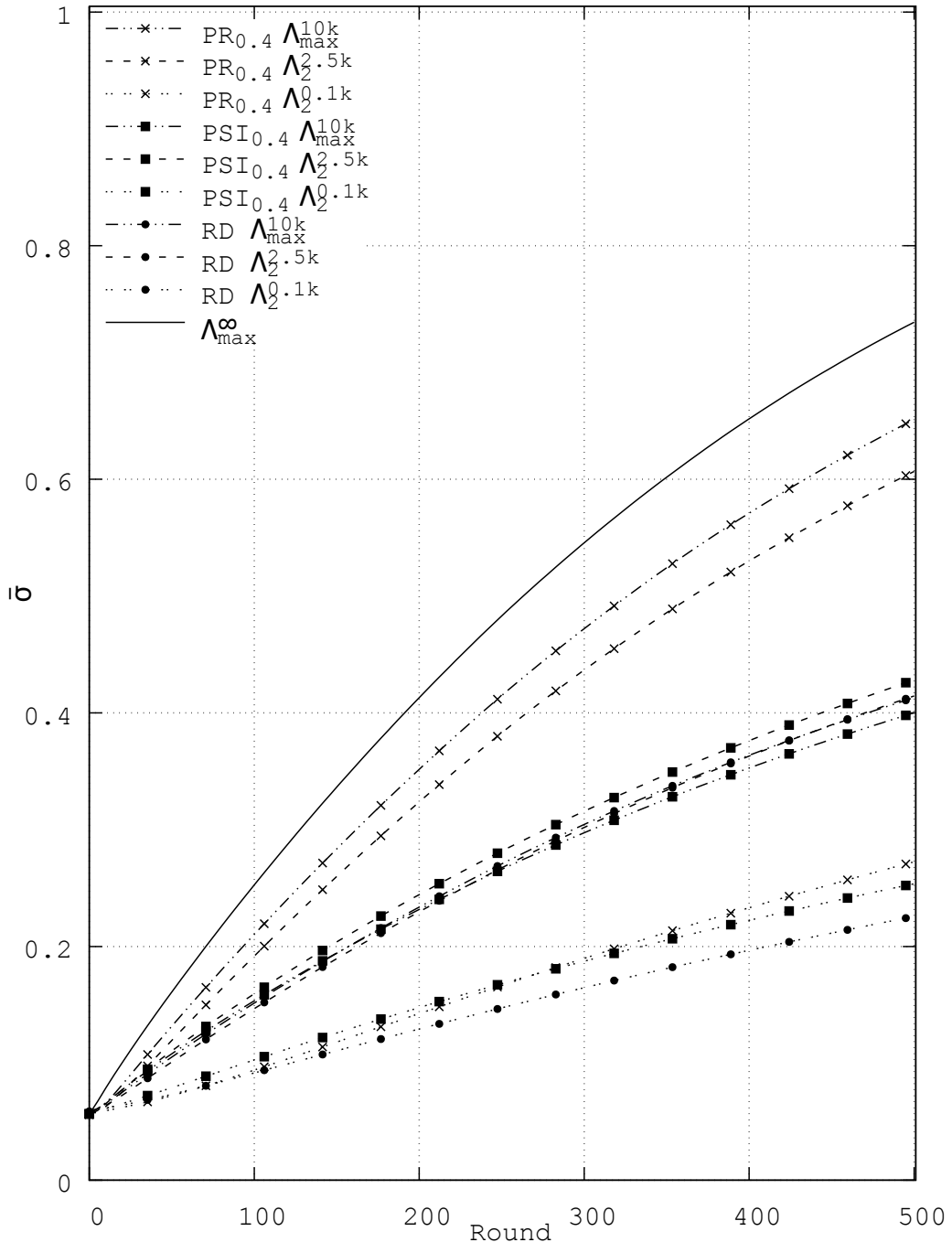


Figure 5.5: *GaussMarkov synthetic mobility: mean global message spread $\bar{\sigma}$ with varied link settings for PSI, Prioritized (PR) and Random (RD) peer syncs, and unlimited link settings for $\Lambda^{\infty}_{\max}$ peer syncs.*

mean global message spread is still higher than their corresponding RD peer-sync results (14% and 11% higher for $\Lambda_2^{2.5k}$ and $\Lambda_{\max}^{10k}$, respectively).

For the GaussMarkov mobility in Figure 5.5, the results for $\text{PSI}_{0.4}$ $\Lambda_2^{0.1k}$ peer syncs in round 500 are approximately in between the results of the $\text{RD}_{0.4}$ $\Lambda_2^{0.1k}$ and the $\text{PR}_{0.4}$ $\Lambda_2^{0.1k}$ peer syncs. Contrarily to the ManhattanGrid mobility results, the GaussMarkov results for $\Lambda_2^{2.5k}$ and $\Lambda_{\max}^{10k}$ peer syncs are only about as high as the RD peer-sync results, and accordingly much lower than the $\text{PR}_{0.4}$ peer-sync results. Similarly to ManhattanGrid, the higher link capacity from $\text{PSI}_{0.4}$ $\Lambda_2^{2.5k}$ to $\Lambda_{\max}^{10k}$ leads to a lower message spread of approximately 3% in the GaussMarkov mobility.

5.2.3 Discussion of the Results

The fact that the PSI peer-sync results are in most of the cases worse than the PR peer-sync results is probably due to the fact that the pool of messages from which the messages are drawn – the group intersections of the encountering node pairs – is more limited than it is in the case of PR syncs. During PR syncs, messages of all the groups the node is a member are eligible to be drawn from until the specified amount of own group messages is reached. During PSI syncs, the set of eligible messages that can be drawn from is smaller, since only messages from groups that both parties are a member of can be selected. Also, groups with a high number of members could potentially monopolize most of the send buffer slots, since these groups have a higher probability to be shared by two nodes.

Therefore, to see whether it would make a difference in terms of message spread, if a fixed number of groups with a high number of members is excluded from the PSI peer syncs, a number of simulations which excluded messages from the largest 8 and 16 groups from being considered in PSI peer syncs have been run. The results for both the empirical and the synthetic mobility datasets were, however, consistent with the findings described above, meaning that it did not have a measurable impact on message spread.

Overall, the use of PSI peer syncs is not justified. The main reason is the observed lower message spread during the simulation runs. However, another important reason to defer from a use of PSI syncs is the required runtime of the PSI protocol. The durations of node encounters may be short in real life, and the PSI calculations could prolong the duration of a peer sync to that extent, that the exchange of messages can not take place anymore due to an interrupted link. Additionally, the nodes can not fully prepare their send buffer before the encounter, because the common group messages can only be selected and put in the send buffer after the PSI protocol has finished, thus further prolonging the PSI peer snyc.

5.3 Conclusion

This chapter summarized the cryptographic primitive of Private Set Intersection and outlined the three classes of protocols available, i.e., Garbled Circuit based protocols, Public Key Cryptography based protocols, and Oblivious Transfer based protocols. Motivated by the microblogging solution's needed properties, a simple Private Set Intersection protocol based on the Diffie-Hellman key exchange was chosen and implemented on smartphones in order to benchmark its performance.

The overall performance was satisfying, so that the protocol was also tested in the simulation. Here, the results were dis-satisfactory, meaning that the simulation runs with PSI peer syncs slightly underperformed the runs with prioritized peer syncs, indicating that PSI peer syncs are generally not beneficial over a strategy based on prioritized selection.

Chapter 6

Privacy-preserving Oblivious RAM Server Syncs

The previous two chapters discussed a peer-to-peer microblogging solution that relies on peer syncs only. This chapter will focus on the server syncs. At first the basic approach to server syncs from Chapter 3 is recapitulated and its shortcomings are stated in Section 6.1, followed by the introduction of Oblivious RAM and its security properties in Section 6.2. A new approach to server syncs using a multi-client Oblivious RAM is given and evaluated in Section 6.3, and Section 6.4 concludes the chapter.

6.1 Basic Approach

In the basic approach, described in Section 3.2.5, the server only functions as a passive data dump from which chunks of messages can be queried and onto which messages can be appended to. In principle, messages can be queried in two ways: the client either downloads only parts of the messages stored on the server. In this case the messages received do not include all messages and the client only gets a partial view of all available messages. The client could also try to download the entire message dump from the server. In this case – depending on the total amount of messages stored on the server – the capacity of the link of the client to the server might be insufficient, the client could encounter a shortage of storage, and the decryption of this amount of messages might become infeasible on a mobile device.

Both mentioned ways to access the server have manifold drawbacks. Since a passive data dump has no intelligence implemented, nodes are not able to query messages for a specific group they are a member of; group-oriented querying of messages is thus not possible. Secondly, many duplicates of messages may reside on the server – since the server must for privacy reasons not have access to any of the decryption keys he cannot remove duplicates. And thirdly, in case of partial access the server can see which part of the message store was accessed – even though the server is not able to infer any group memberships of the clients from these queries, but only the

messages queried by the nodes. (But he may still see that multiple clients access the same set of messages.) To overcome these shortcomings of a basic passive data dump residing on a server, a more advanced solution needs to be found for server-syncs. In this chapter, we present an approach that is based on Oblivious RAMs.

6.2 Oblivious RAM

Informally, an Oblivious RAM (ORAM) enables a client to have read-write access on an encrypted data structure residing on a server in such a way, that the client's access pattern is concealed from the server. This is achieved by continuously shuffling and re-encrypting the data on the server, so that any two access sequences of the same length become computationally indistinguishable. We give a review of the security properties an ORAM has to offer in Section 6.2.1 and then treat related work on single-client ORAMs and multi-client ORAMs in Section 6.2.2 and Section 6.2.3, respectively.

6.2.1 Security Definition of Access Pattern Privacy

We adopt the classic security definition for ORAMs as it is found in [124]. The purpose behind an ORAM is that the server is not able to infer any information from a client's access pattern to a database. Generally, no information leakage from a client's accesses should take place, which particularly includes information such as:

- the data item that is being accessed;
- the last time a data item has been accessed;
- whether the same data item is being accessed (linkability);
- the order in which the data items are accessed, e.g., random or sequential; and
- whether the type of access is a read or a write.

More formally, the following definition of access pattern privacy describes the security properties an ORAM must exhibit.

Definition (Access pattern privacy). *Let $\vec{y}$ be a data request sequence of length L of the form*

$$\vec{y} = ((\mathsf{op}_1, \mathsf{id}_1, \mathsf{dat}_1), (\mathsf{op}_2, \mathsf{id}_2, \mathsf{dat}_2), \ldots, (\mathsf{op}_L, \mathsf{id}_L, \mathsf{dat}_L)),$$

where each op_i denotes a read or write operation, id_i denotes the identifier of the data block being read or written, and dat_i denotes the data written. In this notation, index 1 corresponds to the most recent and index L corresponds to the oldest operation. Let $\mathsf{VIEW}(\vec{y})$ denote the sequence of all messages exchanged between the client and the

remote storage induced by the sequence of data requests $\vec{y}$. An ORAM construction is said to be secure if for any two data requests $\vec{x}$ and $\vec{y}$ of the same length, their views $\mathsf{VIEW}(\vec{x})$ *and* $\mathsf{VIEW}(\vec{y})$ *are computationally indistinguishable (cf. [69]) by anyone but the client.*

6.2.2 Single-client ORAM

First ORAM constructions were initially published by Goldreich [52] and Ostrovsky [90]. Yet, their research questions were different in that they wanted to devise a means of protecting software from piracy. Since then, a sizeable amount of research has been conducted on proposing new and improving existing ORAM constructions [34, 50, 52, 54, 57, 58, 97, 115, 122, 124, 128].

The square root ORAM by Goldreich [52] was the first ORAM construction to be proposed. For this ORAM, the storage on the server consits of $(N + 2\sqrt{N})$ blocks. The first N blocks are encrypted blocks of the database, containing the actual data. Additionally, there is extra storage amounting to $2\sqrt{N}$ blocks which is needed in order to hide the access patterns whenever the same block is accessed more than one time.

Hierarchical ORAM [54] requires the server to layout the data blocks into a hierarchical structure based on levels. On each level the blocks are stored randomly, the position being determined by a randomly selected hash function. From top to bottom, each level contains an increasing number of hash buckets, with each hash bucket containing $\log_2 N$ blocks. In [122] an improved hierarchical ORAM framework is proposed which is based on the use of partitions. Each partition is treated as an ORAM black box that provides a read and write operation whilst hiding the access patterns within that partition. For each partition an actual ORAM construction has to be chosen. In this way, an ORAM based on partitions is built by using this generic construction.

Binary tree ORAM [115] organizes the server storage as a binary tree and every data block is mapped to a leaf node selected uniformly at random – yet, it does not strictly have to be stored at the leaf node. The data block is, however, always placed at one of the nodes that belongs to the path from the root of the tree to the leaf node it is mapped to in the binary tree. A position map is used by the client in order to manage the index of the leaf node each data block is currently mapped to. Path ORAM [124] is an optimized version of the binary tree ORAM.

A comparison of the performance of different ORAMs in [21] shows that Path ORAM is to date the most efficient ORAM construction available.

Note that all mentioned constructions so far are in the client-server model: one client outsources his data to one single server.

6.2.3 Multi-client ORAM

The construction of multi-client ORAMs was initiated by Franz et al. [45], where multiple clients store their data into one ORAM, and give each other access to parts of it. Although this work pioneered further research in the field, it suffers from heavy communication costs, which make the solution difficult to use in practice, particularly in applications where the clients have very limited resources, such as the scenario we are interested in. The work on multi-client ORAMs was further addressed by Williams et al. [131], with the objective to enable access to a server whilst keeping both the file content and the clients' accesses patterns secret. Their major contribution is to support multiple ORAM clients.

Starting with [10], parallel ORAMs began to be further investigated, with milestone works from Boyle et al. [16] and Raykova et al. [78]. These works however, are in a different setting than the one we are interested, as they assume that the *whole data stored* on the remote server is accessible by *every* client. The multi-client constructions of Maffei et al. [79, 80] and Backes et al. [6] focus primarily on anonymizing the queries from the eyes of the server, and leakage that can occur by allowing many clients to access parts of the same data on a server is not dealt with (cf. Section 6.3.1 for the needed multi-client security properties in our setting).

6.3 Multi-client ORAM Server Syncs

In this section, we propose a solution that addresses the issues of the basic approach to server-sync by using a multi-client ORAM to server-sync. It is not surprising that privacy issues arise during the basic approach from Section 6.1 if messages are selectively loaded from the server: for example, the server might learn the groups to which a given client has access to, even if all the messages are encrypted. At the same time, the server might learn – despite the fact that messages are stored encrypted – if a specific message is of more interest than others, simply by observing how many times this particular message has been accessed. If the basic approach is used with the trivial method of simply loading the entire messages stored on the server, it is likely to perform poorly due to inevitable limitations in networking and computation capacities.

In order to remedy the aforementioned privacy leakages and the drawbracks described in Section 6.1, we propose to outsource selected messages of the clients' data in an ORAM data structure. Since all known ORAMs up to this date have to be created with a constant storage size that can not be dynamically increased afterwards, we create an ORAM that accomodates a fixed amount of messages for a specific duration, e.g., a day, a week, or a month, which is subdivided into message timeframes of equal length. We address the above described privacy issues, while at the same

time we do not require the clients to be constantly online, rather they only need to go online in order to read the latest messages, or to write new ones.

The rest of this section is structured as follows. Section 6.3.1 extends the access pattern privacy security definition for the multi-client ORAM scenario, before in Section 6.3.2 the new multi-client ORAM construction for server syncs is presented and its security is elaborated on. Finally, the performance of the ORAM construction is evaluated in Section 6.3.3.

6.3.1 Security Definitions for Multiple Clients

In our setting, multiple clients store their data on an untrusted server. Unfortunately, in such a multi-client scenario, simply plugging an ORAM construction onto the server does not solve all privacy problems. In fact, it is interesting to observe that by doing so, one actually transfers the leakage from the server to the client side: in any ORAM construction, in order to read or write one data block, a client has to access more blocks than the block that the client is actually interested in. Other clients may examine this leaked information in order to gain information on the queried content of a given client.

We work in a setting where every client belongs to various groups, and being a member of a group, the client can read/write *any* block in that particular group. This situation is exemplified in Figure 6.1, that exhibits sample groups, clients and the server hosting the ORAM.

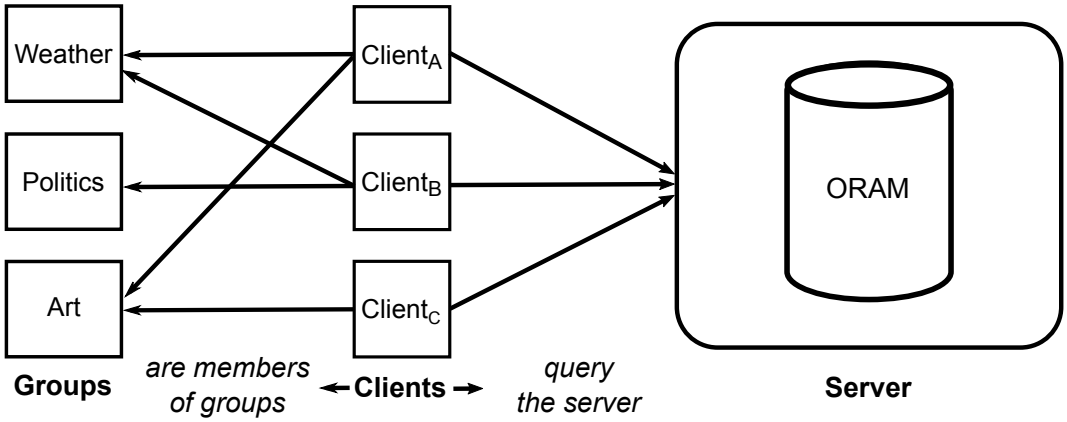


Figure 6.1: *Components of the ORAM-application.*

The security guarantees that we wish to achieve in order to have privacy-preserving server syncs, include making sure that each access to the ORAM – which consists of a read or write operation on one data block of the ORAM – is guaranteed to be private for every client against the server: the server should not be able to tell which block has been accessed, but it also should not be able to tell if a block from a particular group has been accessed. In total, the privacy of the access pattern, i.e., the sum of

all individual accesses to the ORAM, has to be guaranteed for every client against the server. In this thesis, we do not address the problem of client anonymity, i.e., the server can potentially tell whether client A or client B has accessed the system. However, as noted above, the server should not be able to tell whether a client is a member of a particular group or not. Secondly, we want to guarantee that access pattern privacy is achieved for clients who do not belong to the same groups. That is, if client A belongs to the groups 'Weather' and 'Art' and he accesses a message block from 'Art', client B belonging to the groups 'Weather' and 'Politics' should not be able to tell that an access to the group 'Art' has been performed. If however client A had accessed a message block from 'Weather' (clients' A and B common group), client B would have noticed it, but we do not treat this as leakage, as both clients have complete access to that particular group (cf. Figure 6.1). (Note that such leakage cannot be avoided, as each client may notice accesses by observing the content of the blocks.)

In order to be able to argue about the aforementioned security properties, we have to define them in a more precise way. For this, we adjust the definition of access pattern privacy that is common in the ORAM literature, as it has been introduced in Section 6.2.1:

Definition 1 (View). *For a data request sequence of length L of the form*

$$\vec{y} = \left(\left(\mathsf{op}_1, \mathsf{id}_{x_1}^{G_{i_1}}, \mathsf{dat}_{x_1} \right), \left(\mathsf{op}_2, \mathsf{id}_{x_2}^{G_{i_2}}, \mathsf{dat}_{x_2} \right), \ldots, \left(\mathsf{op}_L, \mathsf{id}_{x_L}^{G_{i_L}}, \mathsf{dat}_{x_L} \right) \right),$$

where op_i is a read or write operation, id_{x_i} is the identifier queried for, G_i is the group to which the message block with identifier id_{x_i} belongs, and dat_{x_i} is the data written, we call the $\mathsf{VIEW}(\vec{y})$ (or access pattern) induced by $\vec{y}$, the sequence of all messages exchanged between the client and the remote storage given the sequence of data requests $\vec{y}$.

Definition 2 (Access Pattern Privacy against Server). *A multi-client ORAM is said to guarantee access pattern privacy against a semi-honest server, if for any two data request sequences $\vec{x}$ and $\vec{y}$, the views $\mathsf{VIEW}(\vec{x})$ and $\mathsf{VIEW}(\vec{y})$ are computationally indistinguishable by anyone, but the client who performed them.*

Definition 3 (Access Pattern Privacy against Clients). *A multi-client ORAM is said to guarantee access pattern privacy for a client c against semi-honest, non-colluding clients g, if for any two data request sequences $\vec{x}$ and $\vec{y}$ of c, on groups that the clients g are not members of, the views $\mathsf{VIEW}(\vec{x})$ and $\mathsf{VIEW}(\vec{y})$ are computationally indistinguishable by anyone, but the client c who performed them.*

6.3.2 Construction

In our scenario, we are limited by the computational capabilities of the clients involved. Since the client in all ORAM constructions we are aware of is heavily involved

in the ORAM protocol, we need to choose an ORAM that does not demand expensive computations from the client. The most efficient construction known to date is Path ORAM of Stefanov et al. [124], which we use as a building block in our construction. We thus briefly describe Path ORAM in the following, then we present our privacy-preserving multi-client Path ORAM and discuss how we integrate it in our microblogging architecture, before we elaborate on its security.

Path ORAM

In Path ORAM a client stores N data blocks of size B on a server, in a binary tree structure of height $T = \lceil \log_2 N \rceil$. Each node of the tree stores a constant amount Z of blocks, all encrypted under a semantically secure encryption scheme. Every block is uniquely identified by an identifier $\mathsf{id} \in \{1, \ldots, N\}$. There exist real blocks containing actual data and fake blocks that are assigned with no value, i.e, an encryption of '0'. In the beginning, all blocks in the tree are initialized as fake blocks. Every block is mapped to a leaf of the tree, and this mapping is written in a so-called *position map* by the client. A read (or write) operation for a block is performed by the client by downloading all the blocks along the path from the root of the tree to the leaf indicated in the position map. The block's identifier is then randomly remapped to another leaf in the position map, and the client decrypts all blocks found in the downloaded path. For all these blocks, the client checks its corresponding leaf in the position map, and moves it (if there is enough available space) to the node in the path which is closest to the leaf level and belongs both to the downloaded path and the path to the leaf given by the position map (a process called *eviction*). If a block does not fit anywhere in the downloaded path, then an extra storage called *'stash'* is used by the client to store this overflowing block locally. The blocks found in the stash are also examined during every read (or write) operation and checked if they can be evicted from the stash and placed in the tree[1]. Once the client has examined all blocks found in the downloaded path and in the stash, the client re-encrypts all the blocks in the path and uploads it to the server.

The protocol's security guarantee comes from the following observation: the server's view consists of the leaf, the downloaded path (which the server sends to the client) and the uploaded path (which the server receives from the client). Then, since the leaf is randomly chosen by the client in every access, and the downloaded and uploaded paths are of exactly the same size, encrypted under a semantically secure encryption scheme, any two views are computationally indistinguishable in the eyes of the server.

[1] Since the stash must be stored locally by the client, the stash's size should be reasonably small; in fact, in the paper's full version [121], the authors show that for any access pattern, the probability that the stash exceeds a size of $O(\log N)$ is negligible.

Multi-client Path ORAM

Our privacy-preserving solution for server-syncs needs an ORAM which is able to accomodate multiple groups. A new multi-client ORAM construction based on Path ORAM is in the following outlined. For simplicity, we call it PREaMBLe as in [68]. We consider multiple *groups* in which clients of the system store their message blocks. For every group, the server holds N message blocks. Each of these message blocks is uniquely identified by a specific timeframe. For example, the server will store approximately 2^{13} message blocks per group, given that each day is divided in 1024 timeframes of 85 seconds. Each message block can store up to Z messages (cf. Path ORAM description above). Message block with $\mathsf{id} = 1$ will correspond to the first timeframe (say Monday at time 00:00:00) and the message block with $\mathsf{id} = 2^{13}$ will correspond to the last timeframe (say Sunday at time 23:58:35). For each of these groups, the messages are encrypted using a homomorphic, public key encryption scheme that allows re-randomizations of the ciphertexts without knowledge of the public key (cf. Section 3.2.2). Each group's messages are stored in an individual Path ORAM structure, as described in Section 6.3.2. The blocks of each individual Path ORAM are encrypted under a public/private key pair, which is known only to the clients who are members of that group. The individual Path ORAMs of all groups are concatenated together per node, forming the ORAM tree of our architecture (a small example of the overall structure of the concatenated Path ORAMs containing three groups with $N = 4$ messages is shown in Figure 6.2).

Observe that, since the same number of message blocks is used for every group, this concatenation results in a binary tree of the same height as each of the individual Path ORAM trees. For every group, a position map is created and again all the position maps are concatenated, as in the PREaMBLe-tree.

A client accesses a group's message block (for reading or writing) in the same way a classical Path ORAM is accessed. Here however, since in every node there are also stored message blocks from other groups, which are not accessible by the

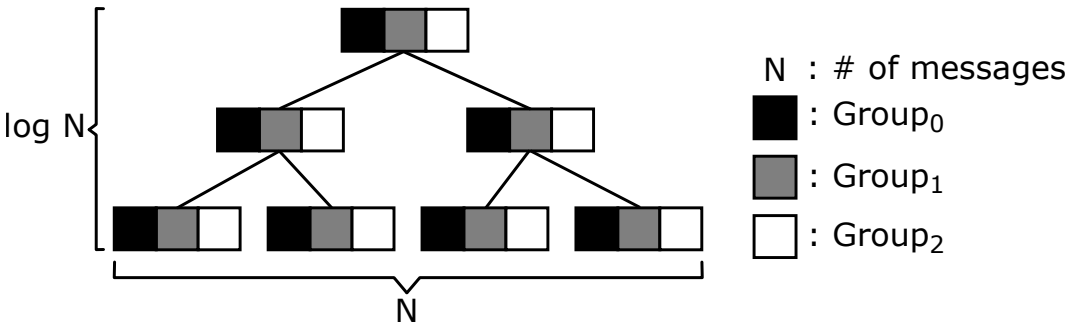


Figure 6.2: PREaMBLe *containing multiple groups, each of them storing N messages.*

client, some extra steps have to be added in the protocol. The client first consults the group's position map to find the leaf to which the queried message block – as queried by its id that corresponds to a uniquely identified timeframe – is mapped. The client then downloads the path to that leaf and decrypts the message blocks of the group he is interested in – observe that this is easily doable, since the message blocks of each group occupy fixed positions within every node of the PREaMBLe-tree. In one of these message blocks the client will find all messages of the requested time frame he is asking for. The client then performs the eviction algorithm (as in the classical Path ORAM), *only for the message blocks of the group he is looking at.* After the eviction is done, the client re-encrypts all the message blocks, that are in now plaintext in the path, and re-randomizes all the message blocks he could not decrypt – note that during this step, homomorphic encryption that allows re-randomization without knowledge of the public key is required (cf. Section 3.2.2). The client finally sends the updated path to the server.

Since accessing the PREaMBLe-tree is in essence an access to a Path ORAM tree, the use of a stash cannot be avoided. Observe that the same identifier of individual groups will be mapped to different leaves, e.g., id_x of group 'Politics' is mapped in leaf y_0, while with very high probability, id_x of group 'Weather' is mapped to leaf y_1 (where of course $y_0 \neq y_1$). Thus, each group will have a stash containing different identifiers. Note also, that since more clients are members of a group, storing the stash locally on the client is not possible. We solve this problem, by fixing the stash size (something that can be done, by using the maximum stash size results from [124]), filling it up with fake message blocks, and storing it encrypted per group on the server. This way, every time a client accesses a message block on the server, he downloads the concatenated group stashes and decrypts the one corresponding to his group in order to use it for performing his eviction. At the end of the access, he re-encrypts all elements of the group stash he operated on, re-randomizes all other stashes, and uploads the entire stash to the server.

It is also important to describe how the position map is accessed. In the original Path ORAM construction, the position map is stored recursively in smaller Path ORAMs, using an idea first described in [123], until a position map of constant size is reached. The same idea can be applied in our construction as well: all the intermediate position maps Path ORAM trees can be concatenated together; in every access to the position map, the client decrypts the nodes he can, and before uploading the path to the server, the client re-randomizes all nodes in every path.

From the above description it is easy to see that, since PREaMBLe is a concatenation of Path ORAM trees, the communication and computation complexity will be similar to those of Path ORAM. Thus for a number G of message groups, a client needs to have a total of $G \cdot O(\log N)$ space available, in order to store the

stashes during every query. The communication and computation complexities are $G \cdot O(\log^2 N)$, thus linear in the amount of message groups available.

Integration of Multi-client Path ORAM for Server Syncs

So far, we introduced the Path ORAM cryptographic primitive as building block and showed how we adapted it in order to be able to host multiple groups. This section deals with the integration of the multi-client Path ORAM in the microblogging architecture, so that users can server-sync utilizing the new ORAM construction. The overall architecture with the ORAM server component can be seen in Figure 6.3.

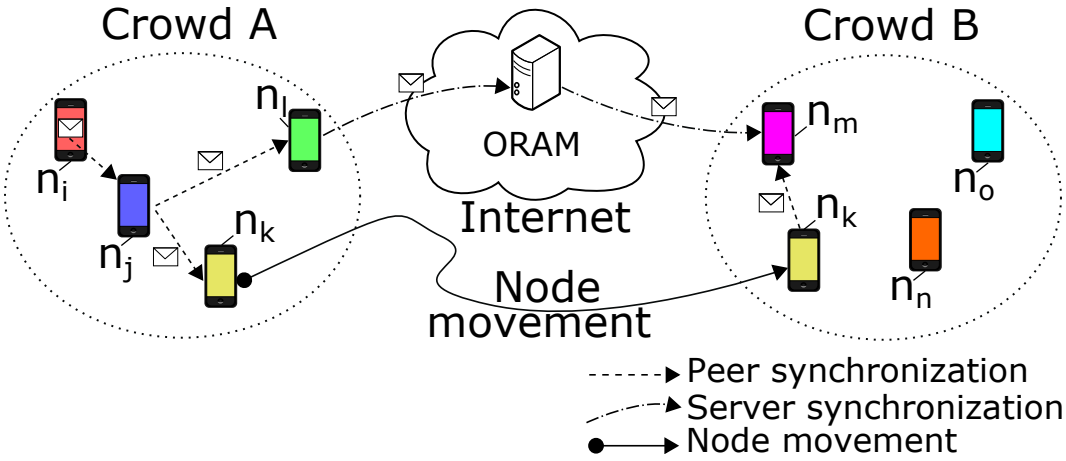


Figure 6.3: *Schematic overview of the architecture with the added ORAM server component.*

As stated, an ORAM has to be of a fixed size and the communication and computation complexities of the multi-client Path ORAM increase linearly with the number of groups it hosts (cf. Section 6.3.3 for an evaluation). Therefore, it is not practical to operate one monolithic ORAM that hosts the messages of all groups for a prolonged period of time. Instead, multiple ORAM instances need to be run on the server with each one containing a certain amount of groups, e.g., 10, 200 and 500, and accomodating the messages created over a uniquely specified period of time, e.g., a day, a week, a month. Periodically, new ORAMs thus need to be created. Subsequently, we discuss the setup, the access, and the privacy implications of this use of multiple ORAMs.

Setup: For simplicity, let us assume that the setup policy is to weekly create ORAMs that each contain a fixed number of groups. Each day is divided in 1024 timeframes of 85 seconds and message block with $\mathsf{id} = 1$ will correspond to the first timeframe on Monday at time 00:00:00 and message block with $\mathsf{id} = 2^{13}$ will correspond to the last timeframe on Sunday at time 23:58:35. The users know this policy and can identify messages by their specific timeframe id. The missing information is which groups will reside on each ORAM. Prior to the setup of the next week's ORAMs,

the server is in an advertisement phase during which it maintains a list of groups that are to be included for the upcoming ORAMs to be created. The list is made of ElGamal encryptions of the neutral element 1, encrypted under the public group keys of the participating groups. Starting with an empty list, it is sequentially expanded. The first user queries the empty list from the server. For each of the groups he is a member he decides if he wants to add it or not. If he wants to add a group, he appends an ElGamal encryption of the element 1 encrypted under the respective public key of the group to the list. He then re-uploads the list to the server. Each subsequent user adds new groups to the list in the following way: firstly, he decides on the groups he wants to add. Then he queries the list from the server, and checks by brute force for each item on the list, if it decrypts to 1 utilizing the private key of any of the groups he wants to add. In case of success, the group is already present in the list and does not need to be added again. In order for the server to be unable to link which item has been added by the user to what position in the list, the user first re-randomizes all existing encryptions of 1, then shuffles the obtained items in a new random order, and the encryptions of 1 for the remaining groups he wishes to add to the list are inserted in the list at random places (the re-randomization of existing encryptions of 1 does not require knowledge of the respective groups' keys, cf. Section 3.2.2). Finally, the altered list is re-uploaded to the server. At the end of this advertisement phase or after the reach of a threshold number of accepted groups, the list of groups is closed by the server and can not be changed anymore. The server subdivides the closed list into sublists that each contain the fixed number of groups. (Any remaining groups are equally distributed on the created sublists.) For each sublist the server is able to initialize an empty ORAM, so that a distinct number of ORAMs – determined by the fixed number of groups – is created.

Access: A client accesses message blocks by their uniquely identifying timeframe. He first queries the server for the available time periods. Upon the first use of any time period, the client has to download the sublists of all the ORAMs created for it. By brute force for each item on the list he checks, if it decrypts to 1 utilizing the private key of any of the groups he is a member of. After this is accomplished, the client knows which of the groups he is a member of are present in which ORAM in the queried time period and at which position within the PREaMBLe-tree, since the message blocks of each group occupy fixed positions within every node of the tree. At this stage, the specific PREaMBLe-tree that needs to be accessed is known to the user and he can access the message block as described above in the subsection on *Multi-client Path ORAM*.

Privacy: This use of a distinct number of ORAMs each hosting a specified number of groups, however, has privacy implications. Ideal from a privacy point of view would be to have all groups hosted in one ORAM, so that a user accessing this ORAM

could potentially be a member of any one of all existing groups. This potential set of groups decreases for a smaller number of groups hosted on an ORAM, yielding some leakage on possible group memberships for the user. Yet, the computations needed to be performed by a user to access an ORAM increase linearly with the number of groups hosted on it. Therefore, privacy trade-offs have to made so that the ORAMs allow for enough privacy by hosting a sufficiently large number of groups while at the same time still being practical to be accessed by users of computationally constrained mobile devices (cf. Section 6.3.3).

Security

In this section, we analyze the security of PREaMBLe. We show that access pattern privacy is guaranteed both against the server but also against clients who do not belong to the same groups. This we show in the following propositions.

PROPOSITION 1. PREaMBLe *achieves access pattern privacy against a semi-honest server.*

Proof. We first argue about access pattern privacy against the server: for two data request sequences $\vec{x}$ and $\vec{y}$ the server sees $\mathsf{VIEW}(\vec{x})$ and $\mathsf{VIEW}(\vec{y})$. Each of the views consists of tuples of the form

$$(\mathsf{leaf}, \mathsf{Path}_{\mathsf{sent}}, \mathsf{Stashes}_{\mathsf{sent}}, \mathsf{Path}_{\mathsf{got}}, \mathsf{Stashes}_{\mathsf{got}}),$$

where leaf marks the path from the root of the tree, $\mathsf{Path}_{\mathsf{sent}}$ is the path containing encrypted nodes that the server sent, $\mathsf{Stashes}_{\mathsf{sent}}$ are the encrypted stashes the server sent, $\mathsf{Path}_{\mathsf{got}}$ is the path containing freshly re-randomized nodes that the server received from the client, and $\mathsf{Stashes}_{\mathsf{got}}$ are the freshly re-randomized stashes that the client sent. Since the leafs are randomly chosen by the client and are independent of the identifiers in $\vec{x}$, the leafs in the two views are indistinguishable. The paths (both the ones sent from the server and received from the server) are of the same length and encrypted under a semantically secure encryption scheme, thus they are also indistinguishable in the two views. Recall now that the stashes are of the same (fixed) size as we discussed in the previous section, they are sent encrypted from the server, and re-randomized when they are sent back to the server. Then, due to the semantic security of the encryption scheme, the stashes are also indistinguishable in the two views. Consequently, the views are computationally indistinguishable in the eyes of the server. □

Arguing about the access pattern privacy against the client is done in a similar way as with the server. However, extra care has to be taken, since in the views, there will appear message blocks that a semi-honest client will be able to decrypt.

PROPOSITION 2. PREaMBLe *achieves access pattern privacy against semi-honest non-colluding clients.*

Proof. Suppose clients A and E are both members of a PREaMBLe instance, and client A performs two data request sequences of length L

$$\vec{x} = \left\{ \left(\mathsf{op}_1, \mathsf{id}_{x_1}^{G_{i_1}}, \mathsf{dat}_{x_1} \right), \ldots, \left(\mathsf{op}_L, \mathsf{id}_{x_L}^{G_{i_L}}, \mathsf{dat}_{x_L} \right) \right\}$$

and

$$\vec{y} = \left\{ \left(\mathsf{op}_1, \mathsf{id}_{y_1}^{G_{j_1}}, \mathsf{dat}_{y_1} \right), \ldots, \left(\mathsf{op}_L, \mathsf{id}_{y_L}^{G_{j_L}}, \mathsf{dat}_{y_L} \right) \right\},$$

such that E is not member of the groups $(G_{i_1}, \ldots G_{i_L}, G_{j_1}, \ldots, G_{j_L})$. Similar to the proof of the access pattern privacy against the server before, we examine the views induced by these data request sequences. Again the views will be of the form

$$(\mathsf{leaf}, \mathsf{Path}_{\mathsf{sent}}, \mathsf{Stashes}_{\mathsf{sent}}, \mathsf{Path}_{\mathsf{got}}, \mathsf{Stashes}_{\mathsf{got}})$$

and as far as leaf is concerned, the same argumentation as in the case of the server is applicable here. Nonetheless, in $\mathsf{Path}_{\mathsf{sent}}$ and $\mathsf{Path}_{\mathsf{got}}$, there will be message blocks which client E will be able to decrypt. Yet, recall that during the eviction client A does not change the place the blocks are stored within a tree node. Thus, all the message blocks that client E can decrypt, remain in exactly the same positions in both $\mathsf{Path}_{\mathsf{sent}}$ and $\mathsf{Path}_{\mathsf{got}}$ in both views, and so they are indistinguishable in the two views. Note also, that in $\mathsf{Path}_{\mathsf{sent}}$ and $\mathsf{Path}_{\mathsf{got}}$ the paths from the recursive access to the position map will also be found, but with a similar argument, they will also be indistinguishable in the two views. The same holds for the stashes in $\mathsf{Stashes}_{\mathsf{sent}}$ and $\mathsf{Stashes}_{\mathsf{got}}$: client E can decrypt some of them, however they will not have been changed during the two accesses, and so they will be indistinguishable in the two views. The above observations lead us to the conclusion that the two views are indistinguishable in the eyes of client E. □

6.3.3 Evaluation

As we have seen in Section 6.3.2, the computation complexity of PREaMBLe is linear in the amount of groups present in the system, and thus it will be governed by the amount of encryptions, decryptions and re-randomizations that have to be performed in every query. We implemented these necessary operations in *C++* using *arm-g++ 4.9* and *openssl 1.1.0* for the cryptographic library and tested the operations on a *Samsung Galaxy S III*. Table 6.1 shows the estimated time needed for a query in various instantiations of PREaMBLe. We set the plain message size to 128 bytes and used an elliptic curve with 256 bits to encrypt the messages. We allowed 85 second time-frames that resulted in a total of 1024 blocks per group. We instantiated 10, 200 and 500 groups. We chose these values because we argue that the achieved level

ORAM stores messages of a	Number of groups	Time in seconds
day	10	1.94
day	200	3.41
day	500	4.73
week	10	33.66
week	200	58.88
week	500	82.71
month	10	83.76
month	200	146.76
month	500	205.78

Table 6.1: *Evaluation of* PREaMBLe*: estimated time in seconds needed for an access to an ORAM storing messages for a day, a week and a month depending on the number of groups hosted on the ORAM. The day, week (7 days), and month (30 days) are each subdivided in 85-second timeframes resulting in 1 024, 7 168 and 30 720 message blocks, respectively.*

of privacy is sufficient in case of 200 groups hosted on an ORAM (and even better in the case of 500 groups). The number of 10 groups is included for comparison, albeit the privacy would be insufficient with such a small number. For these group numbers, we tested PREaMBLe constructions that store message blocks for a day (1 024 message blocks), for a week (7 168 message blocks) and for a month (30 720 message blocks).

Our estimations show that using 10 groups, the query time ranges from 1.94 s (storing message blocks for a day) to 83.76 s (storing message blocks for a month). When 200 groups are present, the query time ranges from 3.41 s (storing message blocks for a day) to 146.76 s (storing message blocks for a month). Lastly, an instantiation with 500 groups, results in query time ranging from 33.66 s (storing message blocks for a day) to 205.78 s (storing message blocks for a month). The above results show that our scheme can be efficiently implemented, even with larger group numbers.

In order to test the applicability of our solution, we also measured the battery drain in the various cases mentioned above. The results are shown in Figure 6.4, where we see that the power consumption is both dependent on the number of message blocks stored, and the number of groups present. Our measurements showed that even with 500 groups present, storing messages for 30 days in 85 s time-frames, one access to the ORAM server consumed approximately 0.084 % of the battery, thus yielding a practical and efficient construction.

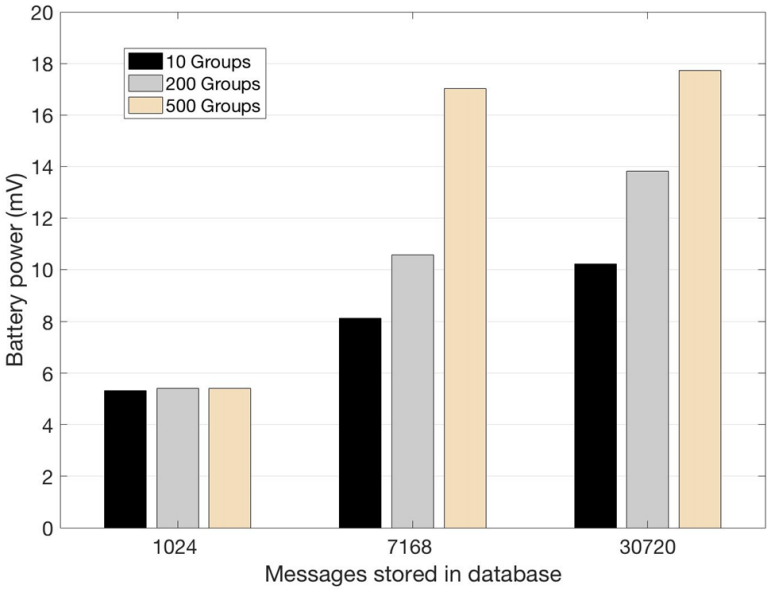


Figure 6.4: *Battery drain due to one access to* PREaMBLe *holding 1 024 message blocks (1 day with 85-second timeframes), 7 168 message blocks (7 days with 85-second timeframes), 30 720 message blocks (30 days with 85-second timeframes), each holding message blocks for 10 groups (left bar), 200 groups (middle bar) and 500 groups (right bar). For instance, the computationally most expensive access with 500 groups and 30 720 message blocks consumed approximately* 0.084 % *of the battery's capacity.*

6.4 Conclusion

To the formerly peer-to-peer network we added a more advanced server component that hosts the concatenation of multiple Path ORAMs, so that clients can go online in order to read messages from or to write messages to the ORAM. Thus the clients are able to privately write messages to and read messages from the server component based on their group memberships and in this way they can achieve a higher message spread in a shorter time span. The ORAM server component added to the formerly pure peer-to-peer network also allows the messages to spread between segmented client clusters (cf. the Lausanne datasets empirical results in Section 4.2.2), while at the same time the clients' privacy is maintained, as the server cannot make any correlation between accesses of different clients. This way, the clients do not disclose their group memberships or the data accessed, neither to the server nor to other clients that do not have access to the same groups.

In total, the solution is practical for this microblogging application and it should be deployed, since clients need only be online when they want to interact with the

server. The server hides the inter-group access patterns and only the intra-group access patterns are seen by the members of singular groups. The communication bounds, the client-side data storage and the client-side computations needed are practical and the evaluation of this new component for ORAM server-syncs showed that it can be run on current mobile devices such as smartphones.

Chapter 7

Conclusion

This thesis introduced a novel form of mobile privacy-enhanced microblogging. The aim of privacy-enhanced microblogging is to allow mobile users to send encrypted microblogging messages in an anonymous, confidential, censorship-resistant and decentralized way.

To that end, at first microblogging was introduced in the context of online social networking, followed by the treatment of related work. Then the microblogging solution was described with its core functional components, including infrastucture and system overview, message format and storage, the used encryption scheme, group and key management, message format and storage, and the devised synchronization mechanisms of peer-syncing and server-syncing.

The devised mobile microblogging system was then thouroughly tested using simulations in its pure peer-to-peer format, which uses only peer syncs alongside with the strategy of prioritized message synchronization – the send buffer contains a fraction of own group messages – and random message synchronization – the send buffer is filled randomly out of all locally stored messages. As input to the simulation served a number of empirical and synthetic mobility datasets. They were used to cover a wide range of mobile user movement patterns and situations. Since user mobility alongside with the used message synchronization strategy are the drivers behind message spread, the resulting message spread differed significantly in the different scenarios tested: the message spread was high in well-connected networks, even if it contained only few nodes, e.g., the Congress datasets, but message spread suffered significantly in the case of network segmentations, e.g., the Lausanne datasets. Generally, a prioritized messages synchronization strategy yielded a better message spread than a random synchronization strategy. Finally, the capacity of the transmission channel was a major limitating factor for message propagation, regardless of the given mobility dataset.

In contrast to well-known anonymity solutions such as crowds and mixes, which specifically address uni-cast routing for session-oriented fixed-line web communica-

tions, privacy-enhanced microblogging uses universal re-encryption instead of a layered one and it is deployed in a delay-tolerant mobile networking scenario. In the purely decentralized system the messages spread without routing overhead using direct communications only (peer syncs). As a result of these design decisions required for anonymity, the microblogging system exhibits a delayed spread of messages on a best effort basis. Yet, these limitations are to be taken into account if the stated security and privacy goals are to be met, as they are with this solution.

To investigate if a synchronization strategy that filled the send buffer with a fraction of common-group messages of the two peer-syncing users yields better results, the use of the cryptographic primitive of Private Set Intersection (PSI) was proposed, and a specific protocol suited for the given application context was chosen and showed to be effectively usable. The simulations were then conducted using PSI peer synchronizations, but showed only mixed results that did not outperform the prioritized peer-syncing simulation runs.

To alleviate the mentioned design limitations of delayed and only best effort message spreading in the purely peer-to-peer based operation of the privacy-enhanced microblogging system, the so far only simple mechanism for server synchronization – with the server as a passive data dump that can not be queried target-orientedly – was replaced with a more sophisticated solution based on an altered version of a Path Oblivious RAM (ORAM) which allowed the users to query their group encrypted messages efficiently and privately, without the server or other users knowing which message has been read or written.

List of Tables

List of Figures

Bibliography

[1] *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*. The Internet Society, 2012. URL: http://www.internetsociety.org/events/ndss-symposium-2012.

[2] Mohammad Shah Alam, Shamim Ara Shawkat, Gontaro Kitazumi, and Mitsuji Matsumoto. Irsimple modeling and performance evaluation for high-speed infrared communications. In *Proceedings of the Global Telecommunications Conference, 2006. GLOBECOM '06, San Francisco, CA, USA, 27 November - 1 December 2006*. IEEE, 2006. URL: http://dx.doi.org/10.1109/GLOCOM.2006.670, doi:10.1109/GLOCOM.2006.670.

[3] Nils Aschenbruck, Raphael Ernst, Elmar Gerhards-Padilla, and Matthias Schwamborn. Bonnmotion: a mobility scenario generation and analysis tool. In L. Felipe Perrone, Giovanni Stea, Jason Liu, Adelinde M. Uhrmacher, and Manuel Villén-Altamirano, editors, *3rd International Conference on Simulation Tools and Techniques, SIMUTools '10, Malaga, Spain - March 16 - 18, 2010*, page 51. ICST/ACM, 2010. URL: http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8684, doi:10.4108/ICST.SIMUTOOLS2010.8684.

[4] Adam J. Aviv, Micah Sherr, Matt Blaze, and Jonathan M. Smith. Evading cellular data monitoring with human movement networks. In Wietse Venema, editor, *5th USENIX Workshop on Hot Topics in Security, HotSec'10, Washington, D.C., USA, August 10, 2010*. USENIX Association, 2010. URL: https://www.usenix.org/conference/hotsec10/evading-cellular-data-monitoring-human-movement-networks.

[5] Dustin Bachrach, Christopher Nunu, Dan S. Wallach, and Matthew K. Wright. #h00t: Censorship resistant microblogging. *CoRR*, abs/1109.6874, 2011. URL: http://arxiv.org/abs/1109.6874.

[6] Michael Backes, Amir Herzberg, Aniket Kate, and Ivan Pryvalov. Anonymous RAM. In Ioannis G. Askoxylakis, Sotiris Ioannidis, Sokratis K. Katsikas, and Catherine A. Meadows, editors, *Computer Security - ESORICS 2016 - 21st European Symposium on Research in Computer Security, Heraklion, Greece, September 26-30, 2016, Proceedings, Part I*, volume 9878 of *Lecture Notes in Computer Science*, pages 344–362. Springer, 2016. URL: http://dx.doi.org/10.1007/978-3-319-45744-4_17, doi:10.1007/978-3-319-45744-4_17.

[7] Kevin S. Bauer, Damon McCoy, Dirk Grunwald, Tadayoshi Kohno, and Douglas C. Sicker. Low-resource routing attacks against tor. In Peng Ning and Ting Yu, editors, *Proceedings of the 2007 ACM Workshop on Privacy in the Electronic Society, WPES 2007, Alexandria, VA, USA, October 29, 2007*, pages 11–20. ACM, 2007. URL: http://doi.acm.org/10.1145/1314333.1314336, doi:10.1145/1314333.1314336.

[8] Yahel Ben-David and Albert Kim. Robin: An attack-resilient microblogging service to circumvent government-imposed communication blackouts, 2012. URL: http://www.eecs.berkeley.edu/~yahel/papers/Robin-An.Attack-resilient.Microblogging.Service.to.Circumvent.Govt.-imposed.Communication.Blackouts.pdf.

[9] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In Simon [117], pages 1–10. URL: http://doi.acm.org/10.1145/62212.62213, doi:10.1145/62212.62213.

[10] Vincent Bindschaedler, Muhammad Naveed, Xiaorui Pan, XiaoFeng Wang, and Yan Huang. Practicing oblivious access on cloud storage: the gap, the fallacy, and the new way forward. In Ray et al. [102], pages 837–849. URL: http://doi.acm.org/10.1145/2810103.2813649, doi:10.1145/2810103.2813649.

[11] Matt Blaze, editor. *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*. USENIX, 2004. URL: https://www.usenix.org/publications/proceedings/?f[0]=im_group_audience{%}3A173.

[12] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970. URL: http://doi.acm.org/10.1145/362686.362692, doi:10.1145/362686.362692.

[13] Oleksandr Bodriagov and Sonja Buchegger. P2P social networks with broadcast encryption protected privacy. In Jan Camenisch, Bruno Crispo, Simone Fischer-Hübner, Ronald Leenes, and Giovanni Russello, editors, *Privacy and Identity Management for Life - 7th IFIP WG 9.2, 9.6/11.7, 11.4, 11.6/PrimeLife International Summer School, Trento, Italy, September 5-9, 2011, Revised Selected Papers*, volume 375 of *IFIP Advances in Information and Communication Technology*, pages 197–206. Springer, 2011. URL: http://dx.doi.org/10.1007/978-3-642-31668-5_15, doi:10.1007/978-3-642-31668-5_15.

[14] Rainer Böhme, George Danezis, Claudia Díaz, Stefan Köpsell, and Andreas Pfitzmann. On the PET workshop panel "mix cascades versus peer-to-peer: Is one concept superior?". In David M. Martin Jr. and Andrei Serjantov, editors, *Privacy Enhancing Technologies, 4th International Workshop, PET 2004, Toronto, Canada, May 26-28, 2004, Revised Selected Papers*, volume 3424 of *Lecture Notes in Computer Science*, pages 243–255. Springer, 2004. URL: http://dx.doi.org/10.1007/11423409_16, doi:10.1007/11423409_16.

[15] Deva K. Borah, Anthony C. Boucouvalas, Christopher C. Davis, Steve Hranilovic, and Konstantinos Yiannopoulos. A review of communication-oriented optical wireless systems. *EURASIP J. Wireless Comm. and Networking*, 2012:91, 2012. URL: http://dx.doi.org/10.1186/1687-1499-2012-91, doi:10.1186/1687-1499-2012-91.

[16] Elette Boyle, Kai-Min Chung, and Rafael Pass. Oblivious parallel RAM and applications. In *TCC 2016-A, Part II*, pages 175–204, 2016.

[17] Jan Camenisch and Gregory M. Zaverucha. Private intersection of certified sets. In Roger Dingledine and Philippe Golle, editors, *Financial Cryptography and Data Security, 13th International Conference, FC 2009, Accra Beach, Barbados, February 23-26, 2009. Revised Selected Papers*, volume 5628 of *Lecture Notes in Computer Science*, pages 108–127. Springer, 2009. URL: https://doi.org/10.1007/978-3-642-03549-4_7, doi:10.1007/978-3-642-03549-4_7.

[18] Tracy Camp, Jeff Boleng, and Vanessa Davies. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing*, 2(5):483–502, 2002. URL: http://dx.doi.org/10.1002/wcm.72, doi:10.1002/wcm.72.

[19] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, and Antony I. T. Rowstron. Scribe: a large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications*, 20(8):1489–1499, 2002. URL: http://dx.doi.org/10.1109/JSAC.2002.803069, doi:10.1109/JSAC.2002.803069.

[20] Augustin Chaintreau, Pan Hui, Jon Crowcroft, Christophe Diot, Richard Gass, and James Scott. Impact of human mobility on opportunistic forwarding algorithms. *IEEE Trans. Mob. Comput.*, 6(6):606–620, 2007. URL: http://dx.doi.org/10.1109/TMC.2007.1060, doi:10.1109/TMC.2007.1060.

[21] Zhao Chang, Dong Xie, and Feifei Li. Oblivious RAM: A dissection and experimental evaluation. *PVLDB*, 9(12):1113–1124, 2016. URL: http://www.vldb.org/pvldb/vol9/p1113-chang.pdf.

[22] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–88, 1981. URL: http://doi.acm.org/10.1145/358549.358563, doi:10.1145/358549.358563.

[23] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In Simon [117], pages 11–19. URL: http://doi.acm.org/10.1145/62212.62214, doi:10.1145/62212.62214.

[24] Chen Chen, Daniele Enrico Asoni, David Barrera, George Danezis, and Adrian Perrig. HORNET: high-speed onion routing at the network layer. In Ray et al. [102], pages 1441–1454. URL: http://doi.acm.org/10.1145/2810103.2813628, doi:10.1145/2810103.2813628.

[25] Ming Ki Chong, Rene Mayrhofer, and Hans Gellersen. A survey of user interaction for spontaneous device association. *ACM Comput. Surv.*, 47(1):8:1–8:40, 2014. URL: http://doi.acm.org/10.1145/2597768, doi:10.1145/2597768.

[26] Nicolas Christin and Reihaneh Safavi-Naini, editors. *Financial Cryptography and Data Security - 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers*, volume 8437 of *Lecture Notes in Computer Science*. Springer, 2014. URL: http://dx.doi.org/10.1007/978-3-662-45472-5, doi:10.1007/978-3-662-45472-5.

[27] Jeremy Clark, Paul C. van Oorschot, and Carlisle Adams. Usability of anonymous web browsing: an examination of tor interfaces and deployability. In Lorrie Faith Cranor, editor, *Proceedings of the 3rd Symposium on Usable Privacy and Security, SOUPS 2007, Pittsburgh, Pennsylvania, USA, July 18-20, 2007*, volume 229 of *ACM International Conference Proceeding Series*, pages 41–51. ACM, 2007. URL: http://doi.acm.org/10.1145/1280680.1280687, doi:10.1145/1280680.1280687.

[28] Aaron Clauset, Cosma Rohilla Shalizi, and Mark E. J. Newman. Power-law distributions in empirical data. *SIAM Review*, 51(4):661–703, 2009. URL: http://dx.doi.org/10.1137/070710111, doi:10.1137/070710111.

[29] Emiliano De Cristofaro, Jihye Kim, and Gene Tsudik. Linear-complexity private set intersection protocols secure in malicious model. In *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 213–231. Springer, 2010. URL: https://doi.org/10.1007/978-3-642-17373-8_13.

[30] Emiliano De Cristofaro, Claudio Soriente, Gene Tsudik, and Andrew Williams. Tweeting with hummingbird: Privacy in large-scale micro-blogging osns. *IEEE Data Eng. Bull.*, 35(4):93–100, 2012. URL: http://sites.computer.org/debull/A12dec/humming.pdf.

[31] Emiliano De Cristofaro and Gene Tsudik. Practical private set intersection protocols with linear complexity. In Radu Sion, editor, *Financial Cryptography and Data Security, 14th International Conference, FC 2010, Tenerife, Canary Islands, January 25-28, 2010, Revised Selected Papers*, volume 6052 of *Lecture Notes in Computer Science*, pages 143–159. Springer, 2010. URL: https://doi.org/10.1007/978-3-642-14577-3_13, doi:10.1007/978-3-642-14577-3_13.

[32] Dana Dachman-Soled, Tal Malkin, Mariana Raykova, and Moti Yung. Efficient robust private set intersection. In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *Applied Cryptography and Network Security, 7th International Conference, ACNS 2009, Paris-Rocquencourt, France, June 2-5, 2009. Proceedings*, volume 5536 of *Lecture Notes in Computer Science*, pages 125–142, 2009. URL: https://doi.org/10.1007/978-3-642-01957-9_8, doi:10.1007/978-3-642-01957-9_8.

[33] Alberto Dainotti, Claudio Squarcella, Emile Aben, Kimberly C. Claffy, Marco Chiesa, Michele Russo, and Antonio Pescapè. Analysis of country-wide internet outages caused by censorship. *IEEE/ACM Trans. Netw.*, 22(6):1964–1977, 2014. URL: https://doi.org/10.1109/TNET.2013.2291244, doi:10.1109/TNET.2013.2291244.

[34] Ivan Damgård, Sigurd Meldgaard, and Jesper Buus Nielsen. Perfectly secure oblivious RAM without random oracles. In Yuval Ishai, editor, *Theory of Cryptography - 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings*, volume 6597 of *Lecture Notes in Computer Science*, pages 144–163. Springer, 2011. URL: https://doi.org/10.1007/978-3-642-19571-6_10, doi:10.1007/978-3-642-19571-6_10.

[35] George Danezis. Wifs'12 keynotes [3 abstracts]. In *Information Forensics and Security (WIFS), 2012 IEEE International Workshop on*, pages 1–4, Dec 2012. doi:10.1109/WIFS.2012.6412614.

[36] Jörg Daubert, Leon Bock, Panayotis Kikiras, Max Mühlhäuser, and Mathias Fischer. Twitterize: Anonymous micro-blogging. In *11th IEEE/ACS International Conference on Computer Systems and Applications, AICCSA 2014, Doha, Qatar, November 10-13, 2014*, pages 817–823. IEEE Computer Society, 2014. URL: http://dx.doi.org/10.1109/AICCSA.2014.7073285, doi:10.1109/AICCSA.2014.7073285.

[37] Karim M. El Defrawy and Gene Tsudik. ALARM: anonymous location-aided routing in suspicious manets. *IEEE Trans. Mob. Comput.*, 10(9):1345–1358, 2011. URL: http://dx.doi.org/10.1109/TMC.2010.256, doi:10.1109/TMC.2010.256.

[38] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Information Theory*, 22(6):644–654, 1976. URL: https://doi.org/10.1109/TIT.1976.1055638, doi:10.1109/TIT.1976.1055638.

[39] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. Tor: The second-generation onion router. In Blaze [11], pages 303–320. URL: http://www.usenix.org/publications/library/proceedings/sec04/tech/dingledine.html.

[40] Changyu Dong, Liqun Chen, and Zikai Wen. When private set intersection meets big data: an efficient and scalable protocol. In Sadeghi et al. [110], pages 789–800. URL: http://doi.acm.org/10.1145/2508859.2516701, doi:10.1145/2508859.2516701.

[41] ETSI-SMG. TR 101 112: Universal Mobile Telecommunications System (UMTS): Selection procedures for the choice of radio transmission technologies of the UMTS (UMTS 30.03 version 3.2.0). Technical report, European Telecommunications Standards Institute (ETSI), 04 1998. URL: http://www.etsi.org/deliver/etsi_tr/101100_101199/101112/03.02.00_60/tr_101112v030200p.pdf.

[42] Hannes Federrath, editor. *Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 25-26, 2000, Proceedings*, volume 2009 of *Lecture Notes in Computer Science*. Springer, 2001.

[43] Ruchith Fernando, Bharat K. Bhargava, and Mark Linderman. Private anonymous messaging. In *IEEE 31st Symposium on Reliable Distributed Systems, SRDS 2012, Irvine, CA, USA, October 8-11, 2012*, pages 430–435. IEEE Computer Society, 2012. URL: http://dx.doi.org/10.1109/SRDS.2012.51, doi:10.1109/SRDS.2012.51.

[44] Emanuel Fonseca, Andreas Festag, Roberto Baldessari, and Rui L. Aguiar. Support of anonymity in vanets - putting pseudonymity into practice. In *IEEE Wireless Communications and Networking Conference, WCNC 2007, Hong Kong, China, 11-15 March, 2007*, pages 3400–3405. IEEE, 2007. URL: http://dx.doi.org/10.1109/WCNC.2007.625, doi:10.1109/WCNC.2007.625.

[45] Martin Franz, Peter Williams, Bogdan Carbunar, Stefan Katzenbeisser, Andreas Peter, Radu Sion, and Miroslava Sotáková. Oblivious outsourced storage with delegation. In George Danezis, editor, *Financial Cryptography and Data Security - 15th International Conference, FC 2011, Gros Islet, St. Lucia, February 28 - March 4, 2011, Revised Selected Papers*, volume 7035 of *Lecture Notes in Computer Science*, pages 127–140. Springer, 2011. URL: http://dx.doi.org/10.1007/978-3-642-27576-0_11, doi:10.1007/978-3-642-27576-0_11.

[46] Michael J. Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword search and oblivious pseudorandom functions. In Joe Kilian, editor, *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, volume 3378 of *Lecture Notes in Computer Science*, pages 303–324. Springer, 2005. URL: http://dx.doi.org/10.1007/978-3-540-30576-7_17, doi:10.1007/978-3-540-30576-7_17.

[47] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2004. URL: https://doi.org/10.1007/978-3-540-24676-3_1.

[48] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Information Theory*, 31(4):469–472, 1985. URL: https://doi.org/10.1109/TIT.1985.1057074, doi:10.1109/TIT.1985.1057074.

[49] Daniel Gatica-Perez, Juha K. Laurila, and Jan Blom. Special issue on the mobile data challenge. *Pervasive and Mobile Computing*, 9(6):751, 2013. URL: http://dx.doi.org/10.1016/j.pmcj.2013.10.001, doi:10.1016/j.pmcj.2013.10.001.

[50] Craig Gentry, Kenny A. Goldman, Shai Halevi, Charanjit S. Jutla, Mariana Raykova, and Daniel Wichs. Optimizing ORAM and using it efficiently for secure computation. In Emiliano De Cristofaro and Matthew K. Wright, editors, *Privacy Enhancing Technologies - 13th International Symposium, PETS 2013, Bloomington, IN, USA, July 10-12, 2013. Proceedings*, volume 7981 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2013. URL: https://doi.org/10.1007/978-3-642-39077-7_1, doi:10.1007/978-3-642-39077-7_1.

[51] Yossi Gilad and Amir Herzberg. Spying in the dark: TCP and tor traffic analysis. In Simone Fischer-Hübner and Matthew K. Wright, editors, *Privacy Enhancing Technologies - 12th International Symposium, PETS 2012, Vigo, Spain, July 11-13, 2012. Proceedings*, volume 7384 of *Lecture Notes in Computer Science*, pages 100–119. Springer, 2012. URL: http://dx.doi.org/10.1007/978-3-642-31680-7_6, doi:10.1007/978-3-642-31680-7_6.

[52] Oded Goldreich. Towards a theory of software protection and simulation by oblivious rams. In Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 182–194. ACM, 1987. URL: http://doi.acm.org/10.1145/28395.28416, doi:10.1145/28395.28416.

[53] Oded Goldreich, Silvio M. Micali, and Avi Wigderson. How to play any mental game. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, pages 218–229, New York, NY, USA, 1987. ACM. URL: http://doi.acm.org/10.1145/28395.28420, doi:10.1145/28395.28420.

[54] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious rams. *J. ACM*, 43(3):431–473, 1996. URL: http://doi.acm.org/10.1145/233551.233553, doi:10.1145/233551.233553.

[55] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Hiding routing information. In Ross J. Anderson, editor, *Information Hiding, First International Workshop, Cambridge, U.K., May 30 - June 1, 1996, Proceedings*, volume 1174 of *Lecture Notes in Computer Science*, pages 137–150. Springer, 1996. URL: http://dx.doi.org/10.1007/3-540-61996-8_37, doi:10.1007/3-540-61996-8_37.

[56] Philippe Golle, Markus Jakobsson, Ari Juels, and Paul F. Syverson. Universal re-encryption for mixnets. In Tatsuaki Okamoto, editor, *Topics in Cryptology - CT-RSA 2004, The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings*, volume 2964 of *Lecture Notes in Computer Science*, pages 163–178. Springer, 2004. URL: http://dx.doi.org/10.1007/978-3-540-24660-2_14, doi:10.1007/978-3-540-24660-2_14.

[57] Michael T. Goodrich and Michael Mitzenmacher. Privacy-preserving access of outsourced data via oblivious RAM simulation. In Luca Aceto, Monika Henzinger, and Jirí Sgall, editors, *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part II*, volume 6756 of *Lecture Notes in Computer Science*, pages 576–587. Springer, 2011. URL: https://doi.org/10.1007/978-3-642-22012-8_46, doi:10.1007/978-3-642-22012-8_46.

[58] Michael T. Goodrich, Michael Mitzenmacher, Olga Ohrimenko, and Roberto Tamassia. Oblivious RAM simulation with efficient worst-case access overhead. In Christian Cachin and Thomas Ristenpart, editors, *Proceedings of the 3rd ACM Cloud Computing Security Workshop, CCSW 2011, Chicago, IL, USA, October 21, 2011*, pages 95–100. ACM, 2011. URL: http://doi.acm.org/10.1145/2046660.2046680, doi:10.1145/2046660.2046680.

[59] Carmit Hazay and Yehuda Lindell. Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In *TCC*, volume 4948 of *Lecture Notes in Computer Science*, pages 155–175. Springer, 2008. URL: https://doi.org/10.1007/978-3-540-78524-8_10.

[60] Carmit Hazay and Kobbi Nissim. Efficient set operations in the presence of malicious adversaries. In *Public Key Cryptography*, volume 6056 of *Lecture Notes in Computer Science*, pages 312–331. Springer, 2010. URL: https://doi.org/10.1007/978-3-642-13013-7_19.

[61] Nicholas Hopper. Challenges in protecting tor hidden services from botnet abuse. In Christin and Safavi-Naini [26], pages 316–325. URL: http://dx.doi.org/10.1007/978-3-662-45472-5_21, doi:10.1007/978-3-662-45472-5_21.

[62] Yan Huang, David Evans, and Jonathan Katz. Private set intersection: Are garbled circuits better than custom protocols? In *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012* [1]. URL: http://www.internetsociety.org/private-set-intersection-are-garbled-circuits-better-custom-protocols.

[63] Bernardo A. Huberman, Matthew K. Franklin, and Tad Hogg. Enhancing privacy and trust in electronic communities. In *EC*, pages 78–86, 1999. URL: http://doi.acm.org/10.1145/336992.337012, doi:10.1145/336992.337012.

[64] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 145–161. Springer, 2003. URL: https://doi.org/10.1007/978-3-540-45146-4_9, doi:10.1007/978-3-540-45146-4_9.

[65] Pierre St. Juste, Heungsik Eom, Benjamin Woodruff, Corey Baker, and Renato J. O. Figueiredo. Enabling decentralised microblogging through p2pvpns. *IJSN*, 8(3):169–178, 2013. URL: http://dx.doi.org/10.1504/IJSN.2013.057699, doi:10.1504/IJSN.2013.057699.

[66] Pierre St. Juste, David Wolinsky, P. Oscar Boykin, Michael J. Covington, and Renato J. O. Figueiredo. Socialvpn: Enabling wide-area collaboration with integrated social and overlay networks. *Computer Networks*, 54(12):1926–1938, 2010. URL: http://dx.doi.org/10.1016/j.comnet.2009.11.019, doi:10.1016/j.comnet.2009.11.019.

[67] Pierre St. Juste, David Wolinsky, P. Oscar Boykin, and Renato J. O. Figueiredo. Litter: A lightweight peer-to-peer microblogging service. In *PASSAT/SocialCom 2011, Privacy, Security, Risk and Trust (PASSAT), 2011 IEEE Third International Conference on and 2011 IEEE Third International Conference on Social Computing (SocialCom), Boston, MA, USA, 9-11 Oct., 2011*, pages 900–903. IEEE, 2011. URL: http://dx.doi.org/10.1109/PASSAT/SocialCom.2011.192, doi:10.1109/PASSAT/SocialCom.2011.192.

[68] Nikolaos P. Karvelas, Marius Senftleben, and Stefan Katzenbeisser. Microblogging in a privacy-preserving way. In *Proceedings of the 12th International Conference on Availability, Reliability and Security, Reggio Calabria, Italy, August 29 - September 01, 2017*, pages 48:1–48:6. ACM, 2017. URL: http://doi.acm.org/10.1145/3098954.3103163, doi:10.1145/3098954.3103163.

[69] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014.

[70] Dogan Kesdogan, Jan Egner, and Roland Büschkes. Stop-and-go-mixes providing probabilistic anonymity in an open system. In David Aucsmith, editor, *Information Hiding, Second International Workshop, Portland, Oregon, USA, April 14-17, 1998, Proceedings*, volume 1525 of *Lecture Notes in Computer Science*, pages 83–98. Springer, 1998. URL: http://dx.doi.org/10.1007/3-540-49380-8_7, doi:10.1007/3-540-49380-8_7.

[71] Lea Kissner and Dawn Xiaodong Song. Privacy-preserving set operations. In *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 241–257. Springer, 2005. URL: https://doi.org/10.1007/11535218_15.

[72] Vladimir Kolesnikov and Ranjit Kumaresan. Improved OT extension for transferring short secrets. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 54–70. Springer, 2013. URL: https://doi.org/10.1007/978-3-642-40084-1_4, doi:10.1007/978-3-642-40084-1_4.

[73] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Efficient batched oblivious PRF with applications to private set intersection. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 818–829. ACM, 2016. URL: http://doi.acm.org/10.1145/2976749.2978381, doi:10.1145/2976749.2978381.

[74] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue B. Moon. What is twitter, a social network or a news media? In Michael Rappa, Paul Jones, Juliana Freire, and Soumen Chakrabarti, editors, *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 591–600. ACM, 2010. URL: http://doi.acm.org/10.1145/1772690.1772751, doi:10.1145/1772690.1772751.

[75] Michael Z. Lee, Alan M. Dunn, Jonathan Katz, Brent Waters, and Emmett Witchel. Anonpass: Practical anonymous subscriptions. *IEEE Security & Privacy*, 12(3):20–27, 2014. URL: http://dx.doi.org/10.1109/MSP.2013.158, doi:10.1109/MSP.2013.158.

[76] Michael Z. Lee, Alan M. Dunn, Brent Waters, Emmett Witchel, and Jonathan Katz. Anonpass: Practical anonymous subscriptions. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 319–333. IEEE Computer Society, 2013. URL: http://dx.doi.org/10.1109/SP.2013.29, doi:10.1109/SP.2013.29.

[77] Adam Lerner, Giulia C. Fanti, Yahel Ben-David, Jesus Garcia, Paul Schmitt, and Barath Raghavan. Rangzen: Anonymously getting the word out in a blackout. *CoRR*, abs/1612.03371, 2016. URL: http://arxiv.org/abs/1612.03371.

[78] Jacob R. Lorch, James W. Mickens, Bryan Parno, Mariana Raykova, and Joshua Schiffman. Toward practical private access to data centers via parallel ORAM. *IACR Cryptology ePrint Archive*, 2012:133, 2012. URL: http://eprint.iacr.org/2012/133.

[79] Matteo Maffei, Giulio Malavolta, Manuel Reinert, and Dominique Schröder. Privacy and access control for outsourced personal records. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 341–358. IEEE Computer Society, 2015. URL: http://dx.doi.org/10.1109/SP.2015.28, doi:10.1109/SP.2015.28.

[80] Matteo Maffei, Giulio Malavolta, Manuel Reinert, and Dominique Schröder. Maliciously secure multi-client oram. *IACR Cryptology ePrint Archive*, 2017. URL: http://eprint.iacr.org/2017/329.

[81] Lior Malka and Jonathan Katz. Vmcrypt - modular software architecture for scalable secure computation. Cryptology ePrint Archive, Report 2010/584, 2010. URL: https://eprint.iacr.org/2010/584.

[82] Dahlia Malkhi, Noam Nisan, Benny Pinkas, and Yaron Sella. Fairplay - secure two-party computation system. In Blaze [11], pages 287–302. URL: http://www.usenix.org/publications/library/proceedings/sec04/tech/malkhi.html.

[83] Rene Mayrhofer and Hans Gellersen. Shake well before use: Intuitive and secure pairing of mobile devices. *IEEE Trans. Mob. Comput.*, 8(6):792–806, 2009. URL: http://dx.doi.org/10.1109/TMC.2009.51, doi:10.1109/TMC.2009.51.

[84] Liam McNamara, Cecilia Mascolo, and Licia Capra. Media sharing based on colocation prediction in urban transport. In J. J. Garcia-Luna-Aceves, Raghupathy Sivakumar, and Peter Steenkiste, editors, *Proceedings of the 14th Annual International Conference on Mobile Computing and Networking, MOBICOM 2008, San Francisco, California, USA, September 14-19, 2008*, pages 58–69. ACM, 2008. URL: http://doi.acm.org/10.1145/1409944.1409953, doi:10.1145/1409944.1409953.

[85] Catherine A. Meadows. A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In *Proceedings of the 1986 IEEE Symposium on Security and Privacy, Oakland, California, USA, April 7-9, 1986*, pages 134–137. IEEE Computer Society, 1986. URL: http://dx.doi.org/10.1109/SP.1986.10022, doi:10.1109/SP.1986.10022.

[86] MobiSocial Computing Laboratory, Stanford University. Stanford MobiSocial Project. https://mobisocial.stanford.edu/ . URL: https://mobisocial.stanford.edu/.

[87] Thyago Mota, Aarti Munjal, and Tracy Camp. Large-scale human mobility analysis based on mobile phone and social media communication: A case-study in africa. In Christian S. Jensen, Xing Xie, Vladimir Zadorozhny, Sanjay Madria, Evaggelia Pitoura, Baihua Zheng, and Chi-Yin Chow, editors, *16th IEEE International Conference on Mobile Data Management, MDM 2015, Pittsburgh, PA, USA, June 15-18, 2015 - Volume 2*, pages 86–91. IEEE Computer Society, 2015. URL: http://dx.doi.org/10.1109/MDM.2015.84, doi:10.1109/MDM.2015.84.

[88] Steven J. Murdoch and George Danezis. Low-cost traffic analysis of tor. In *2005 IEEE Symposium on Security and Privacy (S&P 2005), 8-11 May 2005, Oakland, CA, USA*, pages 183–195. IEEE Computer Society, 2005. URL: http://dx.doi.org/10.1109/SP.2005.12, doi:10.1109/SP.2005.12.

[89] N.N. Schutz der Privatsphäre: “Privacy by Design" als technisches und gesellschaftliches Konstruktionsprinzip (Vorlesungsnr. 02-01-0017-ku). Lecture notes, June 12th, 2013.

[90] Rafail Ostrovsky. Efficient computation on oblivious rams. In Harriet Ortiz, editor, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 514–523. ACM, 1990. URL: http://doi.acm.org/10.1145/100216.100289, doi:10.1145/100216.100289.

[91] Jörg Ott, Esa Hyytiä, Pasi E. Lassila, Tobias Vaegs, and Jussi Kangasharju. Floating content: Information sharing in urban areas. In *Ninth Annual IEEE International Conference on Pervasive Computing and Communications, PerCom 2011, 21-25 March 2011, Seattle, WA, USA, Proceedings*, pages 136–146. IEEE, 2011. URL: http://dx.doi.org/10.1109/PERCOM.2011.5767578, doi:10.1109/PERCOM.2011.5767578.

[92] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999. URL: https://doi.org/10.1007/3-540-48910-X_16, doi:10.1007/3-540-48910-X_16.

[93] Andriy Panchenko, Fabian Lanze, and Thomas Engel. Improving performance and anonymity in the tor network. In *31st IEEE International Performance Computing and Communications Conference, IPCCC 2012, Austin, TX, USA, December 1-3, 2012*, pages 1–10. IEEE Computer Society, 2012. URL: http://dx.doi.org/10.1109/PCCC.2012.6407715, doi:10.1109/PCCC.2012.6407715.

[94] Timothy Perfitt and Burkhard Englert. Megaphone: Fault tolerant, scalable, and trustworthy p2p microblogging. In *2010 Fifth International Conference on Internet and Web Applications and Services*, pages 469–477, May 2010. doi:10.1109/ICIW.2010.77.

[95] Andreas Pfitzmann and Marit Hansen. A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management, 2010. URL: https://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf.

[96] Andreas Pfitzmann and Marit Köhntopp. Anonymity, unobservability, and pseudonymity - A proposal for terminology. In Federrath [42], pages 1–9. URL: http://dx.doi.org/10.1007/3-540-44702-4_1, doi:10.1007/3-540-44702-4_1.

[97] Benny Pinkas and Tzachy Reinman. Oblivious RAM revisited. In Tal Rabin, editor, *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*, pages 502–519. Springer, 2010. URL: https://doi.org/10.1007/978-3-642-14623-7_27, doi:10.1007/978-3-642-14623-7_27.

[98] Benny Pinkas, Thomas Schneider, Gil Segev, and Michael Zohner. Phasing: Private set intersection using permutation-based hashing. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 515–530, Washington, D.C., 2015. USENIX Association. URL: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/pinkas.

[99] Benny Pinkas, Thomas Schneider, Christian Weinert, and Udi Wieder. Efficient circuit-based PSI via cuckoo hashing. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 125–157. Springer, 2018. URL: https://doi.org/10.1007/978-3-319-78372-7_5, doi:10.1007/978-3-319-78372-7_5.

[100] Benny Pinkas, Thomas Schneider, and Michael Zohner. Faster private set intersection based on OT extension. In Kevin Fu and Jaeyeon Jung, editors, *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.*, pages 797–812. USENIX Association, 2014. URL: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/pinkas.

[101] Benny Pinkas, Thomas Schneider, and Michael Zohner. Scalable private set intersection based on OT extension. *ACM Trans. Priv. Secur.*, 21(2):7:1–7:35, 2018. URL: http://doi.acm.org/10.1145/3154794, doi:10.1145/3154794.

[102] Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors. *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015*. ACM, 2015. URL: http://dl.acm.org/citation.cfm?id=2810103.

[103] Jean-François Raymond. Traffic analysis: Protocols, attacks, design issues, and open problems. In Federrath [42], pages 10–29. URL: http://dx.doi.org/10.1007/3-540-44702-4_2, doi:10.1007/3-540-44702-4_2.

[104] Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for web transactions. *ACM Trans. Inf. Syst. Secur.*, 1(1):66–92, 1998. URL: http://doi.acm.org/10.1145/290163.290168, doi:10.1145/290163.290168.

[105] Michael K. Reiter and Aviel D. Rubin. Anonymous web transactions with crowds. *Commun. ACM*, 42(2):32–38, 1999. URL: http://doi.acm.org/10.1145/293411.293778, doi:10.1145/293411.293778.

[106] Injong Rhee, Minsu Shin, Seongik Hong, Kyunghan Lee, and Song Chong. Human mobility patterns and their impact on delay tolerant networks. In Constantine Dovrolis, Vern Paxson, and Stefan Savage, editors, *6th ACM Workshop on Hot Topics in Networks - HotNets-VI, Atlanta, Georgia, USA, November 14-15, 2007*. ACM SIGCOMM, 2007. URL: http://conferences.sigcomm.org/hotnets/2007/papers/hotnets6-final108.pdf.

[107] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978. URL: http://doi.acm.org/10.1145/359340.359342, doi:10.1145/359340.359342.

[108] Michael Rogers and Eleanor Saitta. Secure communication over diverse transports: [short paper]. In Ting Yu and Nikita Borisov, editors, *Proceedings of the 11th annual ACM Workshop on Privacy in the Electronic Society, WPES 2012, Raleigh, NC, USA, October 15, 2012*, pages 75–80. ACM, 2012. URL: http://doi.acm.org/10.1145/2381966.2381977, doi:10.1145/2381966.2381977.

[109] Antony I. T. Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In Rachid Guerraoui, editor, *Middleware 2001, IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg,*

Germany, November 12-16, 2001, Proceedings, volume 2218 of *Lecture Notes in Computer Science*, pages 329–350. Springer, 2001. URL: http://dx.doi.org/10.1007/3-540-45518-3_18, doi:10.1007/3-540-45518-3_18.

[110] Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors. *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*. ACM, 2013. URL: http://dl.acm.org/citation.cfm?id=2508859.

[111] Jan Schejbal. A real-world study of mobile peer-to-peer networking. Master's thesis, TU Darmstadt, 2014. URL: http://www.janschejbal.de/permanent/masterthesis.

[112] Marius Senftleben, Ana Barroso, Mihai Bucicoiu, Matthias Hollick, Stefan Katzenbeisser, and Erik Tews. On the privacy and performance of mobile anonymous microblogging. *IEEE Trans. Information Forensics and Security*, 11(7):1578–1591, 2016. URL: http://dx.doi.org/10.1109/TIFS.2016.2541633, doi:10.1109/TIFS.2016.2541633.

[113] Marius Senftleben, Mihai Bucicoiu, Erik Tews, Frederik Armknecht, Stefan Katzenbeisser, and Ahmad-Reza Sadeghi. MoP-2-MoP - Mobile Private Microblogging. In Christin and Safavi-Naini [26], pages 384–396. URL: http://dx.doi.org/10.1007/978-3-662-45472-5_25, doi:10.1007/978-3-662-45472-5_25.

[114] Andrei Serjantov and Richard E. Newman. On the anonymity of timed pool mixes. In Dimitris Gritzalis, Sabrina De Capitani di Vimercati, Pierangela Samarati, and Sokratis K. Katsikas, editors, *Security and Privacy in the Age of Uncertainty, IFIP TC11 18th International Conference on Information Security (SEC2003), May 26-28, 2003, Athens, Greece*, volume 250 of *IFIP Conference Proceedings*, pages 427–434. Kluwer, 2003. URL: http://dx.doi.org/10.1007/978-0-387-35691-4_41, doi:10.1007/978-0-387-35691-4_41.

[115] Elaine Shi, T.-H. Hubert Chan, Emil Stefanov, and Mingfei Li. Oblivious RAM with o((logn)3) worst-case cost. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, volume 7073 of *Lecture Notes in Computer Science*, pages 197–214. Springer, 2011. URL: https://doi.org/10.1007/978-3-642-25385-0_11, doi:10.1007/978-3-642-25385-0_11.

[116] Reza Shokri, George Theodorakopoulos, Carmela Troncoso, Jean-Pierre Hubaux, and Jean-Yves Le Boudec. Protecting location privacy: optimal strategy against localization attacks. In Yu et al. [136], pages 617–627. URL: http://doi.acm.org/10.1145/2382196.2382261, doi:10.1145/2382196.2382261.

[117] Janos Simon, editor. *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*. ACM, 1988.

[118] Indrajeet Singh, Michael Butkiewicz, Harsha V. Madhyastha, Srikanth V. Krishnamurthy, and Sateesh Addepalli. Twitsper: Tweeting privately. *IEEE Security & Privacy*, 11(3):46–50, 2013. URL: http://dx.doi.org/10.1109/MSP.2013.3, doi:10.1109/MSP.2013.3.

[119] Yair Sovran, Alana Libonati, and Jinyang Li. Pass it on: social networks stymie censors. In Adriana Iamnitchi and Stefan Saroiu, editors, *Proceedings of the 7th international conference on Peer-to-peer systems, IPTPS'08, Tampa, FL, USA, February 25-26, 2008*, page 3. USENIX, 2008. URL: http://www.iptps.org/papers-2008/73.pdf.

[120] Mudhakar Srivatsa and Mike Hicks. Deanonymizing mobility traces: using social network as a side-channel. In Yu et al. [136], pages 628–637. URL: http://doi.acm.org/10.1145/2382196.2382262, doi:10.1145/2382196.2382262.

[121] Emil Stefanov and Elaine Shi. Path O-RAM: an extremely simple oblivious RAM protocol. *CoRR*, abs/1202.5150, 2012. URL: http://arxiv.org/abs/1202.5150.

[122] Emil Stefanov, Elaine Shi, and Dawn Xiaodong Song. Towards practical oblivious RAM. In *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012* [1]. URL: http://www.internetsociety.org/towards-practical-oblivious-ram.

[123] Emil Stefanov, Elaine Shi, and Dawn Xiaodong Song. Towards practical oblivious RAM. In *NDSS*, 2012.

[124] Emil Stefanov, Marten van Dijk, Elaine Shi, Christopher W. Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path ORAM: an extremely simple oblivious RAM protocol. In Sadeghi et al. [110], pages 299–310. URL: http://doi.acm.org/10.1145/2508859.2516660, doi:10.1145/2508859.2516660.

[125] Jing Su, Alvin Chin, Anna Popivanova, Ashvin Goel, and Eyal de Lara. User mobility for opportunistic ad-hoc networking. In *6th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2004), 2-10 December 2004, Lake District National Park, UK*, pages 41–50. IEEE Computer Society, 2004. URL: http://dx.doi.org/10.1109/MCSA.2004.29, doi:10.1109/MCSA.2004.29.

[126] Jorma T. Virtamo, Esa Hyytiä, and Pasi E. Lassila. Criticality condition for information floating with random walk of nodes. *Perform. Eval.*, 70(2):114–123, 2013. URL: http://dx.doi.org/10.1016/j.peva.2012.11.001, doi:10.1016/j.peva.2012.11.001.

[127] Heribert Vollmer. *Introduction to Circuit Complexity: A Uniform Approach*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1999.

[128] Xiao Shaun Wang, Yan Huang, T.-H. Hubert Chan, Abhi Shelat, and Elaine Shi. SCORAM: oblivious RAM for secure computation. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, pages 191–202. ACM, 2014. URL: http://doi.acm.org/10.1145/2660267.2660365, doi:10.1145/2660267.2660365.

[129] Samuel D. Warren and Louis D. Brandeis. The right to privacy. *Harvard Law Review*, 4(5):193–220, December 1890.

[130] Jianshu Weng, Ee-Peng Lim, Jing Jiang, and Qi He. Twitterrank: finding topic-sensitive influential twitterers. In Brian D. Davison, Torsten Suel, Nick Craswell, and Bing Liu, editors, *Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010, New York, NY, USA, February 4-6, 2010*, pages 261–270. ACM, 2010. URL: http://doi.acm.org/10.1145/1718487.1718520, doi:10.1145/1718487.1718520.

[131] Peter Williams, Radu Sion, and Alin Tomescu. Privatefs: a parallel oblivious file system. In Yu et al. [136], pages 977–988. URL: http://doi.acm.org/10.1145/2382196.2382299, doi:10.1145/2382196.2382299.

[132] Wireless Lab, Shanghai Jiao Tong University. SUVnet-trace data. http://wirelesslab.sjtu.edu.cn. URL: http://wirelesslab.sjtu.edu.cn.

[133] Bing Wu, Jianmin Chen, Jie Wu, and Mihaela Cardei. *A Survey of Attacks and Countermeasures in Mobile Ad Hoc Networks*, pages 103–135. Springer US, Boston, MA, 2007. URL: http://dx.doi.org/10.1007/978-0-387-33112-6_5, doi:10.1007/978-0-387-33112-6_5.

[134] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 162–167. IEEE Computer Society, 1986. URL: http://dx.doi.org/10.1109/SFCS.1986.25, doi:10.1109/SFCS.1986.25.

[135] Xun Yi, Russell Paulet, and Elisa Bertino. *Private Information Retrieval*. Synthesis Lectures on Information Security, Privacy, and Trust. Morgan & Claypool Publishers, 2013. URL: http://dx.doi.org/10.2200/S00524ED1V01Y201307SPT005, doi:10.2200/S00524ED1V01Y201307SPT005.

[136] Ting Yu, George Danezis, and Virgil D. Gligor, editors. *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*. ACM, 2012. URL: http://dl.acm.org/citation.cfm?id=2382196.

[137] Yanchao Zhang, Wei Liu, Wenjing Lou, and Yuguang Fang. MASK: anonymous on-demand routing in mobile ad hoc networks. *IEEE Trans. Wireless Communications*, 5(9):2376–2385, 2006. URL: http://dx.doi.org/10.1109/TWC.2006.1687761, doi:10.1109/TWC.2006.1687761.

Anhang A

Erklärung

Hiermit erkläre ich, dass ich die Arbeit – abgesehen von den in ihr ausdrücklich genannten Hilfen – selbstständig verfasst habe.

Darmstadt, den 17. April 2018